MASTER EN SCIENCES INFORMATIQUES

Accès aux bibliothèques digitales via les Personal Digital Assistants

Sadek-Omar, Samir

Award date:
2002

Abstract

The purpose of the proposed thesis is the evaluation of the possibilities of taking existing digital library technologies and using it for educating those who normally do not have access to the Internet. Beyond this goal, we will examine how it is possible to access and find information in the World Wide Web by using the *Personal Digital Assistants*.

We have built an interface that allows *Personal Digital Assistants* users to access the content of the Greenstone digital library by using the technologies of the AvantGo compagny.

For this thesis we will go through the effects of the small screen when we are searching for information in the World Wide Web. We will also give an insight of the solutions proposed by a few existing systems.

In the light of the above, an assessment will be made of the ability offered by our system to access digital libraries. But especially this study will allow us to evaluate our system as a general means for accessing and searching information in the World Wide Web.

Keywords: Personal Digital Assistants (PDAs), World Wide Web, searching, browsing, keywords, digital library, computer-human interaction.

Résumé

Le but de ce mémoire est d'évaluer la possibilité d'utiliser les technologies des bibliothèques digitales pour éduquer ceux qui n'ont pas accès à l'Internet. Au delà de cet objectif nous verrons comment il est possible d'accéder et de trouver de l'information dans le World Wide Web par l'intermédiaire des *Personal Digital Assistants*.

Nous avons construit une interface qui permet aux utilisateurs des *Personal Digital Assistants* d'accéder au contenu des bibliothèques digitales de Greenstone en utilisant les technologies de la compagnie AvantGo.

Ici, nous examinerons quelles sont les impacts de la recherche d'informations dans le World Wide Web lorsqu'on utilise des petits écrans. Nous verrons quelles sont les solutions proposées par quelques systèmes pour pallier les difficultés.

A la lumière de ceci, nous pourrons évaluer la solution qu'offre notre système pour l'accès aux bibliothèques digitales. Mais surtout, cette étude nous permettra de juger la pertinence de ce système comme moyen d'accès et de recherche d'information dans le World Wide Web.

Mots-clés: Personal Digital Assistants (PDAs), World Wide Web, recherche, navigation, mots clés, bibliothèque digitale, interaction homme-machine.

Acknowledgements (Remerciements)

Mes remerciements les plus sincères s'adressent bien entendu à mes parents et à ma famille, mais aussi à mes amis. Ces cinq années n'ont pas toujours été faciles, souvent pour moi, et parfois pour eux. Alors, merci pour leur précieux soutien et leur indispensable présence.

Au Professeur Jacques Berleur s.j pour ses conseils et son soutien. Je le remercie également pour l'opportunité qu'il m'a offerte de vivre une expérience de travail magnifique à l'Université de Waikato en Nouvelle-Zélande.

To the Professor Matt Jones, from the University of Waikato, for his guidance and advice during our training. Also for the time he gave me during four mounths. I would like to thank him, as well as his family and friends, for their warm welcome in this so wonderful country.

To the whole staff of the University of Waikato, professors and secretaries, for their kindness and warm receptions. And of course to all the people who allow me to have a so nice stay in New Zealand. Special thanks go to: John McPherson, Stuart Yeates, François Paradis and Stefan boddy who helped me every time I needed it.

Finalement, à toute l'Université de Namur, pour l'accueil et l'enseignement qu'ils m'ont procurés pendant cinq années. Ainsi qu'à tous mes professeurs qui, depuis le début, ont contribué à notre éducation.

Table des matières

Table des figures

Liste des tableaux

Introduction

Objectifs du projet GreenstoneToAvantgo

GreenstoneToAvantgo est le nom du projet que nous avons mené à l'Université de Waikato (Nouvelle-Zélande). Son but est de permettre aux PDAs de visualiser le contenu des bibliothèques digitales de Greenstone grâce aux technologies d'accès à l'Internet mises à la disposition des utilisateurs par la compagnie AvantGo. En d'autres termes, GreenstoneToAvantgo est une interface entre les bibliothèques digitales de Greenstone et les technologies propriétaires de la compagnie AvantGo.

Greenstone est une suite de software[1] développée par l'Université de Waikato dans le cadre du projet «New Zealand Digital Library Project» (NZDLP). Il permet de construire des bibliothèques digitales et de les rendre accessibles sur l'Internet ou sur CD-ROM. L'ambition de NZDLP est de solliciter les utilisateurs, et particulièrement les universités, bibliothèques, et autres institutions de services publiques pour construire leur propre bibliothèque digitale.

Greenstone est développé et distribué en coopération avec l'UNESCO[2] et Human Info NGO[3] dans le but de favoriser la dissémination d'informations scientifiques et culturelles à travers le monde, et particulièrement aux pays en voie de développement. A ce titre, 14 collections d'informations humanitaires sont disponibles sur le site de la bibliothèque digitale de Nouvelle-Zélande[4].

A terme, GreenstoneToAvantgo doit permettre aux étudiants, chercheurs, etc., d'accéder aux bibliothèques digitales de Greenstone par le biais de leur PDA. Et à plus court terme aux 14 collections humanitaires du site de Nouvelle-Zélande.

[1] Greenstone is open-source issued under the terms of the GNU General Public License.
[2] UNESCO (United Nations Educational, Scientific and Cultural Organization)
[3] The Human Info NGO est basé à Anvers (Belgique)
[4] `http:\\www.nzdl.org`

Ce projet devrait bénéficier du développement des réseaux de télécommunication sans fil dans nos pays mais également dans les pays en voie de développement. En effet, en Afrique par exemple, le «boom» du téléphone mobile est un phénomène qui se manifeste clairement. Et des indices laissent penser qu'avec l'intensification de la concurrence, le développement des télécommunications s'accélera ([ITU TELECOM, 2001]).

Comme nous le verrons par la suite, les nouveaux PDAs sont capables de se connecter aux réseaux mobiles et d'avoir accès aux World Wide Web n'importe où et à n'importe quel moment, tant que les opérateurs de téléphonie mobile (faisant office de fournisseur d'accès à l'Internet) le permettent.

Le projet GreenstoneToAvantgo s'inscrit également dans les sytèmes existants d'accès au World Wide Web. En effet, l'explosion des technologies sans fil a largement accru le besoin de mettre au point des solutions d'accès par le biais des mobiles et des PDAs.

Problématiques: PDA & WWW

L'accès au World Wide Web signifie plus concrètement la recherche d'informations diverses. C'est ce problème qui tiendra lieu de fil rouge le long du mémoire.

Quel que soit le médium utilisé pour la recherche d'informations, un certain nombre de contraintes sont invariables. Si l'on veut maximiser les chances qu'ont les utilisateurs de découvrir l'information qu'ils recherchent, il est indispensable de les identifier. Néammoins, les contraintes spécifiques aux PDAs attirerons plus particulièrement notre attention.

La recherche d'informations impose deux types de problématiques. L'une est technique et inspire la question suivante: comment rendre disponible l'information du World Wide Web sachant que les PDAs ont de peu de bande passante et peu de mémoire ? L'autre est relative à l'interaction homme-machine et se formalise comme suit: quelles sont les exigences auxquelles il faut répondre pour aider l'utilisateur à trouver l'information qu'il recherche sur le World Wide Web ?

La première problématique nous contraint à mettre au point des solutions techniques. Nous pensons par exemple à l'utilisation d'un proxy qui effectuerait des traitements sur les pages Web avant de les renvoyer aux PDAs. Un traitement typique est la compression.

La deuxième problématique implique l'identification du comportement des internautes quand ils recherchent de l'information sur le World Wide Web via leur PDA. Mais naturellement, elle s'inscrit également dans un contexte général de recherche d'informations quel que soit le médium utilisé.

Ces deux problématiques sont corrélées. Afin de proposer un système d'accès efficace au World Wide Web pour PDAs, il est indispensable de mettre en relation l'apect technique avec l'aspect humain. Il faut donc proposer un système qui pallie les difficultés techniques et qui mette l'utilisateur au coeur du processus de conception.

Nous entendons par «système efficace» (pour l'accès au World Wide Web via les PDAs), l'ensemble des technologies qui maximisent les chances qu'ont les utilisateurs de trouver l'information qu'ils recherchent. Nous identifions entre autres le navigateur Web pour PDA.

Notons que notre démarche ne s'inscrit nullement dans l'accès du World Wide Web pour un but récréatif. Il sagit plutôt d'aider les internautes à trouver les informations dont ils ont réellement besoin.

Objectif du mémoire

Le but de ce mémoire est d'évaluation le projet; et de situer cette solution par rapport à l'ensemble des systèmes d'accès pour PDAs proposés actuellement.

Nous présenterons ce mémoire en montrant d'une part les moyens disponibles pour répondre aux contraintes techniques intrinsèques des PDAs; et d'autre part les éléments qui déterminent les contraintes humaines. Nous examinerons plus particulièrement l'influence de la taille de l'écran des PDAs sur les méthodes de recherche et l'aptitude d'assimilation de l'information des utilisateurs.

De plus, nous verrons quelques principes déjà mis en oeuvre pour les navigateurs conventionnels[5]. Ceci nous permettra de distinguer les difficultés liées aux contraintes des PDAs des difficultés relatives au World Wide Web tout simplement. Nous verrons que les principes établis antérieurement aux PDAs reste d'actualité dans le cadre de nos recherches.

[5]Nous distinguons les navigateurs conventionnels des navigateurs pour PDA car ils disposent d'un écran de taille normale et d'une connection à l'Internet munie d'une plus large bande passante que les navigateurs pour PDAs

Contenu du mémoire

Ce mémoire se divise en deux parties. La première s'attelle à répondre à la problématique de recherche d'informations dans le WWW par le biais des PDAs; la seconde a pour objet de décrire le système GreenstoneToAvantgo.

La première Partie se compose comme suit:

Le Chapitre 1 décrit les éléments qui déterminent les contraintes relatives aux interactions entre les utilisateurs et leur PDA. Nous verrons entre autres les éléments constitutifs d'un PDA, mais également comment il est possible de maximiser l'espace d'affichage disponible pour les applications. Nous examinerons les impacts qu'a un petit écran sur la lecture et la compréhension d'un texte, ainsi que les opérations qu'effectuent les utilisateurs sur l'écran pour accomplir différentes tâches.

Dans le Chapitre 2 nous verrons de manière générale les difficultés liées à l'accès et la recherche d'informations par les PDAs. Nous examinerons quelles sont les raisons qui feront des navigateurs Web pour PDAs l'application de l'Internet de demain; ainsi que les approches préconisées pour l'accès au WWW. Nous apprendrons que trois éléments sont sensibles pour la recherche d'informations: la navigation, la recherche par mots clés, et l'introduction de données dans le PDA.

Le Chapitre 3 nous permettra d'identifier le comportements qu'ont les utilisateurs pour trouver de l'information par le biais du PDA. Nous examinerons à travers une expérimentation, comment ils s'y prennent pour effectuer un certain nombre de tâches dans le World Wide Web lorsqu'ils disposent d'un petit écran. Nous dégagerons des principes de design pour mettre en oeuvre un système de navigation efficace spécifique aux PDAs. Nous apprendrons que le design des systèmes de liens hypertextes n'est pas un problème trivial, et qu'un certain nombre d'enseignements sont de mises dans le contexte des PDAs. Et finalement, nous présenterons trois exemples d'implémentation mettant en oeuvre les principes découverts dans ce chapitre: WebTwig, Power Browser et Knowledge Agent Bases.

Dans le Chapitre 4 nous mettrons en lumière les différents avantages que l'on peut tirer des techniques de résumé généré automatiquement. Nous examinerons les bases des systèmes d'Information Retrieval (IR) dont la qualité a un impact direct sur la recherche dans l'Internet. Les procédés de génération automatique de résumés sont indissociables de l'IR, et nous en verrons quelques uns. Nous examinerons ensuite comment les principes vus peuvent être utilisés dans le World Wide Web; notamment par l'exemple d'OCELOT, un

prototype qui génére automatiquement des résumés de pages Web. Nous dégagerons certains enseignements sur les potentiels offerts par le Web dans ce cadre. Et finalement, à l'aide d'un exemple, nous discuterons la manière d'améliorer les sytèmes de navigation pour PDAs.

Dans la seconde Partie de ce mémoire nous nous focaliserons sur le système GreenstoneToAvantGo. Elle se compose comme suit:

Dans le Chapitre 5 nous présenterons le software Greenstone. Nous verrons en quoi les bibliothèques digitales peuvent aider les pays en voie de développement; ainsi que l'organisation de la bibliothèque digitale de Greenstone. Nous mettrons en évidence un certain nombre d'exigences auxquelles répond Greenstone et nous examinerons les fonctions de recherche d'informations qui sont mises à notre disposition. Nous finirons ce chapitre en présentant le processus de construction de bibliothèque digitale, et jetterons un regard au système Greenstone du point de vue de son architecture.

Finalement, nous présenterons dans le Chapitre 6 l'application GreenstoneToAvantgo. Nous présenterons nos motivations et expliquerons son fonctionnement et son implémentation.

Première partie

Recherche d'informations grâce aux PDAs

Chapitre 1

Interaction Homme-PDA

Le but de ce chapitre est de décrire les éléments qui déterminent les contraintes relatives aux interactions entre les utilisateurs et leur PDA. Nous décrirons dans la Section 1.1 les éléments constitutifs d'un PDA. Nous verrons également dans la Section 1.2 comment il est possible de maximiser l'espace d'affichage disponible pour les applications. Et finalement, nous examinerons dans la Section 1.3 les impacts qu'a un petit écran sur la lecture et la compréhension d'un texte, ainsi que les opérations qu'effectuent les utilisateurs pour accomplir différentes tâches.

1.1 Description du Palm

C'est intentionnellement que nous décrirons le Palm (voir Figure 1.1) car c'est sur lui que les différents tests ont été effectués dans le projet GreenstoneToAvantgo. Il est d'ailleurs largement répandu sur le marché mondial (En 2000, Palm représentait 55.9 % du marché des PDAs[1]).

Il existe deux types de PDA: ceux basés sur le très célébre Windows CE de Microsoft tel que le système d'exploitation Pocket PC implémenté par de nombreux constructeurs (e.g. Casio, Compaq, Hewlett Packard); et ceux basés sur le Palm OS tels que Palm, Sony, HandSpring, Nokia,...

La taille du Palm est généralement[2] de 12 $cm \times 8\ cm \times 1.5\ cm$ pour un

[1] source: `http://www.palmos.com/licensees/palm/`

[2] Nous rappelons que les métriques fournies le sont uniquement à titre indicatif.

FIG. 1.1 – *PDA de type Palm.*

poids d'environ 160 g. On peut aisément idéntifier trois composants:

1. le rebord contenant un menu de 6 boutons permettant d'accéder directement à certaines fonctions[3] du stystème tel que l'utilisation du bloc-notes.
2. une partie se trouvant dans le bas de l'écran, prenant moins d'un tiers de la surface totale affichable, est divisée en deux:
   (a) au centre se trouve un emplacement destiné à la reconnaissance de l'écriture: avec une partie située à gauche pour la reconnaissance des lettres, et à droite la reconnaissance des chiffres.
   (b) autour de l'emplacement décrit au point 2(a) on trouve un second menu permettant l'accès à d'autres fonctions par le biais de 4 icônes.
3. la partie affichable, proprement dite, est destinée aux applications: elle a une largeur et une longueur de 6 cm (15 lignes peuvent être affichées) pour une taille de 610×610 pixels. Certains modèles offrent un écran couleur.

La limite de lisibilité des caractères pour la plupart des utilisateurs se trouve entre 9 et 12 points ([Kamba et al., 1996]):

Comme je descendais des Fleuves impassibles,
Je ne me sentais plus guidé par les haleurs :
Des Peaux-Rouges criards les avaient pris pour cibles
Les ayant cloués nus aux poteaux de couleurs.

Arthur Rimbaud, Le Bateau Ivre (1871)

1.1.1 Interface d'input

Les PDAs offrent principalement deux interfaces à l'introduction de données dont la reconnaissance d'écriture par stylus[4], qui appliqué à l'écran tactile, est capable de saisir le mouvement de la main et de le traduire dans son correspondant: une lettre, un chiffre ou un caractère spécial. Nous pouvons

[3] Afin d'avoir des explications supplémentaires sur les fonctions du système voir `http://www.palm.com`

[4] Petit bâton en plastic, légèrement pointu, conçu pour les PDAs servant à l'interaction avec l'écran tactile afin d'utiliser toutes les fonctions proposées, e.g. sélectionner un objet de l'interface utilisateur (comme une icône servant à lancer une application), introduire des données, ou faire défiler une page Web.

deviner le mode de fonctionnement de reconnaissance d'écriture des lettres en regardant la Figure 1.2. Chaque lettre a une et une seule manière d'être formée. L'Utilisateur n'a donc qu'à mémoriser le graphisme propre à chaque caractère.

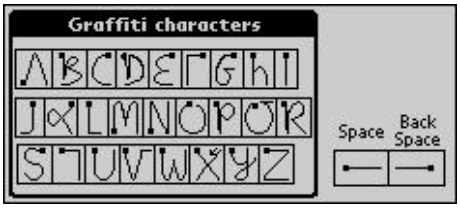

FIG. 1.2 – *Aperçu des caractères possibles d' effectuer.*

Un autre moyen est le «soft keyboard»: il s'agit d'un clavier virtuel intégré dans l'écran du PDA, et qui présente les mêmes touches qu'un clavier minimum standard (voir Figure 1.3). Il vient se placer sur l'espace d'affichage déstiné aux applications. Pour saisir une lettre, il suffit — à l'aide de son stylus — d'appuyer sur le caractère désiré.

FIG. 1.3 – *Clavier.*

1.2 Interface utilisateur

Le paradigme du bureau auquel nous sommes familiers n'est pas de mise dans le contexte du PDA. La plupart des activités effectuées quotidiennement sont

possibles par l'intermédiaire d'un environnement comprenant des icônes, de multiples fenêtres,... Bien évidemment, ces genres de procédés ne sont pas souhaitables sur le PDA. Outre le fait qu'ils utiliseraient trop d'espace, leur utilisation serait vite impraticable. Cela est sans doute une des raisons pour lesquelles la reconnaissance d'écriture a été inventée : éviter l'utilisation d'un clavier trop encombrant([Kamba et al., 1996]).

1.2.1 Maximiser l'espace d'affichage

Comme l'a dit Rob Haitani «product manager» pour la première génération de Palm Pilot (cité dans [Tambe, 2000]):«*quickly finds that every pixel counts and that there's just no room. Any embellishment adds more pixels. Simply adding a border to an application implies adding one pixel for the border itself and a margin of three or four pixels between the border and the text. That's five pixels on four sides. With a 160*160 pixel screen, this is 12% of the total screen size.*»

Bien que les éléments de l'interface doivent être discrets, il n'en reste pas moins qu'ils doivent être suffisamment larges pour pouvoir être sollicités par un stylus ou encore par un doigt. Comme nous le voyons sur la Figure 1.4, la présence de tels objets pour une simple application peut considérablement diminuer l'espace d'affichage.

Minimisier l'importance des menus

Les menus sont largement utilisés en Interfaces Hommes Machines (IHM): ils fournissent de l'information sur les fonctions des applications et la possibilité de les utiliser. C'est pour ces raisons qu'un certain nombre de personnes ont cherché à minimiser la place employée par les menus, tout en leur garantissant un bonne ergonomie.

Menu pop-up ([Kurtenbach and Buxton, 1994]) Ce système dipose d'un menu qui apparaît sous le stylus quand il est maintenu sur l'écran environ 1/3 seconde (voir Figure 1.5). Le menu a une forme de tarte dans laquelle chaque part représente un sous-menu (ou une fonction) pouvant être sélectionné en le traversant avec le stylus. Une alternative, pour les utilisateurs familiers, est de traverser l'écran dans le sens de l'item que l'on voudrait sélectionner si le menu était présent. Cela économise le temps d'attente pour l'apparation

FIG. 1.4 – *L'espace d'affichage contient une simple application.*

du menu. En résumé, la sélection est basée sur la direction du mouvement appliqué sur l'écran.

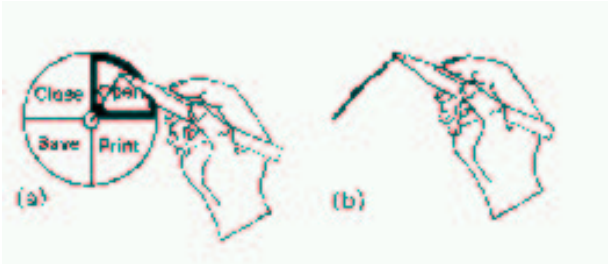

FIG. 1.5 – *Menu pop-up.*

Un certain nombre d'études ont été faites afin d'apprécier le nombre de fonctions maximum, ainsi que la profondeur des sous-menus, que l'on peut avoir sur le menu. Lorsque l'utilisateur emploie souvent les mêmes fonctions, il n'a plus besoin d'attendre l'apparition du menu; dans ce cas le menu est très efficace. Cependant cette méthode est inutile lorsque le menu change dynamiquement.

Menu semi-transparent ([Kamba et al., 1996]) Cette approche a pour but d'empêcher la disparition d'une partie de l'écran derrière le menu. L'idée est d'offrir un menu semi-transparent (voir Figure 1.6). Conceptuellement, l'écran est divisé en deux couches: l'une contenant le menu et l'autre les informations utiles. Le passage d'une couche à l'autre dépend de la durée de l'interaction que l'utilisateur entretient avec l'écran en posant son stylus dessus. L'efficacité du système dépend de la visibilité du menu et du temps d'attente pour passer d'une couche à l'autre.

L'utilisation d'un zoom

Le zooming user Interfaces (ZUI) est devenu très populaire depuis qu'il a été introduit par Ken Perlin et David Fox quand ils ont développé le Pad à l'Université de New York. Le Pad++ par exemple (une amélioration du Pad), est une interface graphique basée sur le zoom: les objets peuvent être placés n'importe où et dans n'importe quelle taille. L'utilisateur peut les voir dans la taille désirée, en appliquant le zoom sur l'objet de son choix. Ce système est utilisé pour naviguer au sein de l'interface et ainsi permettre à l'utilisateur de voir de nombreuses informations. Il offre maintes facilités

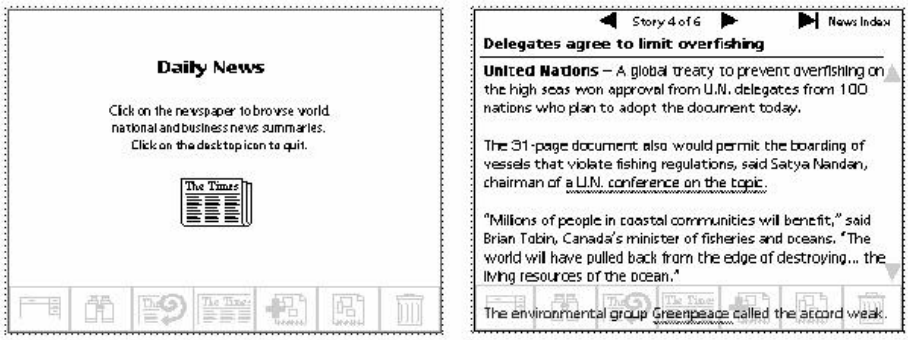

FIG. 1.6 – *Menu semi-transparent.*

ergonomiques pour la navigation afin d'éviter que l'utilisateur ne se perde ([Bederson and Hollan, 1995]).

1.3 L'Aquisition d'informations sur PDA

Les études sur l'impact des écrans de taille réduite sur le comportement de l'utilisateur, ont principalement été effectuées entre les années 1980 et le début des années 1990. La plupart d'entre elles analysent les effets sur deux facteurs: lecture et compréhension de la variation du nombre de lignes visibles (hauteur), ou de la variation du nombre de caractères visibles par ligne (largeur).

Compréhension et lecture

Les études de Ducknicky et Kolers ont montré que la hauteur minimum d'une bonne compréhension d'un texte est de 4 lignes. Si la hauteur est de 20 lignes la compréhension n'est pas améliorée significativement, mais la lecture est 9% plus rapide. Si la largeur est divisée par 3, les performances de la lecture sont nettement plus mauvaises: elle ralentit de 25% (cité dans [Jones et al., 1999a]).

Dillon et al. ont effectué une étude portant sur 3500 textes, dont le but était d'estimer l'aptitude des utilisateurs à en résumer les points essentiels. Ils ont montré que la compréhension des utilisateurs qui disposaient d'un écran n'affichant que 20 à 60 lignes, était aussi bonne que ceux disposant d'un écran large ([Jones et al., 1999a]).

Schneiderman a mené une étude qui exigeait l'utilisation de liens hypertextes. Deux groupes fûrent constitués afin de se soumettre à des tests de compréhension: naviguer à travers les textes en utilisant les liens hypertextes, et répondre à des questions. L'un des groupe disposait d'un écran de 18 lignes et l'autre de 34 lignes. Aucune différence significative de temps n'a été enregistrée pour terminer les tests (cité dans [Jones et al., 1999a]).

Finalement, Schneiderman a montré que les utilisateurs possédant un écran de 22 lignes prenait 9.2 minutes pour lire un texte et répondre à des questions, alors que ceux disposant de 60 lignes prenait 7.8 minutes (une amélioration de 15%)(cité dans [Jones et al., 1999a]).

Interaction écran-utilisateur

Les études de Dillon ont mis en lumière deux effets
(cité dans [Jones et al., 1999a]):

1. Les utilisateurs du petit écran interagissent beaucoup plus avec l'écran que ceux disposant d'un grand écran: ils montent et descentent sur les pages précédentes afin, sans doute, d'offrir plus de contexte à leur lecture;
2. 75% des utilisateurs faisant l'objet de l'expérience auraient préféré travailler sur un écran plus large. Même si les performances ne sont objectivement pas mauvaises, il semblerait que les utilisateurs perçoivent le système comme étant moins bon.

Il est intéressant de se demander ce qu'il en est de la capacité à utiliser un menu (e.g. ceux des «home pages» de plusieurs sites Web par exemple). Swierengo a montré que si la hauteur tournait autour de 12–24 lignes, il n'y avait alors absolument aucun problème (cité dans [Jones et al., 1999a]).

1.4 Conclusions

L'attention que nous portons aux métriques vues aux Sections 1.2 et 1.3 tient au fait qu'elles nous permettent d'estimer à priori les qualités d'affichage des PDAs pour l'accomplissement de certaines tâches. Concrètement, nous voulons savoir si la taille de l'affichage des PDAs ne sera pas un frein à la recherche d'informations sur l'Internet.

Dans l'ensemble les résultats de la Section 1.3 sont positifs: la taille de l'écran n'a pas de grandes répercussions sur les aptitudes qu'ont les utilisateurs pour lire et comprendre un texte, tant qu'un minimum de lignes et une largeur raisonnable soient affichés. En effet, la vitesse de lecture est quelque peu ralentie tandis que la compréhension reste plus ou moins la même.

De plus nous avons vu que les utilisateurs du petit écran interagissaient beaucoup plus avec l'écran. Ils devraient donc utiliser plus souvent la barre de défilement de leur navigateur, sur leur PDA, pour lire et comprendre les pages Web. Mais cela ne devrait pas poser de problème: à l'époque le fait d'utiliser la barre de défilement sur les navigateurs conventionnels, pour voir l'entiéreté d'une page, était considéré comme un défaut de design. Aujourd'hui, selon Nielsen, 10% des utilisateurs vont toujours faire défiler la page, et en général, l'utilisateur ne sera pas dérangé de le faire si necessaire. Cet état de fait est sans doute dû à l'habitude qu'ils ont pris sur le World Wide Web ([Nielsen, 1999a]). Nous ponvons donc présumer que l'utilisation de la barre de défilement, à l'instar des navigateurs conventionnels, ne devrait pas poser de poblème à l'internaute experimenté sur PDA.

Chapitre 2

Accéder à l'Internet via le PDA

Ce chapitre a pour but de décrire de manière générale les difficultés liées à l'accès et la recherche d'informations sur l'Internet par les PDAs. Nous examinerons dans la Section 2.2 quelles sont les raisons qui feront des navigateurs Web pour PDAs l'application de l'Internet de demain. Nous verrons dans la Section 2.3 quelles sont les approches préconisées pour l'accès au WWW des PDAs. Et finalement, dans les Section 2.4 et 2.5 nous verrons que trois éléments sont sensibles pour la recherche d'informations: la navigation[1], la recherche par mots clés, et l'introduction de données dans le PDA.

2.1 Introduction

Les potentiels offerts par les progrès technologiques en matière d'infrastructure de réseaux digitaux sans fil ont massivement accru le nombre de connexions. Les opérateurs sont sans cesse en train de développer des standards et des services de téléphonie mobile (e.g. le GSM [Global System for Mobile telecommunications] en Europe et en Asie). C'est ainsi qu'au début des années 1990s grâce aux systèmes de télécommunication de 1re Génération ils ont pu toucher un grand marché. Aujourd'hui les innovations des réseaux de 2.5e (GPRS/EDGE) et de 3e (UMTS) Génération supportent le trafic par paquets (au sens de IP[Internet Protocol]) et augmentent la bande passante disponible pour l'utilisateur final. Le but de l'UMTS étant de mettre en place un réseau multimédia capable de délivrer aux mobiles (PDAs, téléphones sans fil, ...) un large panel de services de données.

[1] Le terme «naviguer» signifie: passer de pages Web en pages Web en utilisant les liens hypertextes qui les relient.

Au niveau hardware, les PDAs sont plus fins, plus légers, plus rapides, utilisent moins d'énergie et produisent moins de chaleur, dotés de plus en plus de mémoire et offrant de plus en plus de capacités de stockage. On compte aussi un plus grand éventail de techniques en matière d'input de données (e.g. clavier, stylus, son, caméra) et d'output (e.g.video).

2.2 Les Navigateurs pour PDAs: futur «Internet killer app»

Au niveau software, les industriels ne se sont pas fait attendre pour le développement d'applications portables sur PDAs: allant de la simple calculatrice en passant par les suites courantes de traitement de textes jusqu'aux applications spécifiques à certains métiers. Toutefois, l'application que l'on présente aujourd'hui comme la prochaine «Internet killer app» est le navigateur Web pour PDA. Cet état de fait résulte, en grande partie, de l'intégration d'un modem sans fil comme standard d'usine([Nielsen, 1999b]). Certains pensent que le World Wide Web sera accédé dans le futur plus fréquemment par les mobiles que par les ordinateurs conventionnels ([Jones et al., 2000]). Les possibilités d'utilisation sont aussi diverses que variées: achats en ligne, consultation de business news, réservation, voire gestion de compte bancaire.

La précédente «application tueuse de l'Internet» a été (et est toujours) le courrier électronique (courriel) et ce pour deux raisons. La première est qu'il répond à un besoin clé pour les consommateurs: celui de communiquer et de rester en contact avec les autres. En effet, le courriel offre un moyen simple et rapide de prolonger une rencontre ou de maintenir une relation professionnelle. Et la deuxième raison se retrouve dans l'*interaction que l'utilisateur établit avec l'application*: il saisit facilement les concepts sous-jacents à la création, à l'envoi et à la réception d'un message ([Jones et al., 2000]).

Il va s'en dire que le besoin d'accès à l'Internet n'importe où, à n'importe quel instant est un fait, suite aux exigences professionnels d'obtetenir des informations fraîches, mises à jour en temps réel et accessibles dès que l'utilité s'en fait sentir. (D'ailleurs, la vente de PDAs dans le monde, pour l'année 2002, est estimée à 14,8 million d'unités, dont 80% achetés par des entreprises et des particuliers pour des applications professionnelles [Blyaert, 1999]). Mais aussi dû à l'effort du marketing pour nous convaincre d'emporter notre bureau et les informations concernant nos loisirs partout où nous allons.

Pour ce qui est de la facilité d'utilisation, il n'y a qu'un seul mot à dire:

«Insuffisante». Les raisons sont variées et complexes. C'est ce que nous verrons par la suite.

2.3 Deux approches possibles pour l'accès à l'Internet

Bien que les PDAs aient évolué et qu'ils évolueront encore, il n'en reste pas moins que la taille de l'affichage et la résolution ne changeront pas. C'est tout à fait normal étant donnée la nature de l'objet. Ces deux facteurs (taille et résolution) ont une importance non négligeable sur la manière dont les personnes acquièrent l'information.

Dans les dernières années beaucoup d'applications pour PDAs ont été développées sans référence au facteur CHI (Computer Human Interaction), fabriquant ainsi des produits techniquement viables mais dont la qualité d'utilisation était médiocre. Pour prendre un exemple, considérons un navigateur Web pour PDA disposant d'un espace d'affichage réduit. S'il accède à un site Web conçu pour un écran couleur d'au minimum 640x480 de résolution — voir plus —, l'aire d'affichage sur le PDA sera divisée par quatre, ce qui rendra le site inesthétique et même complétement illisible.

Afin d'accèder au Web grâce aux PDAs deux types de solution ont été développées: l'adaptation des sites et des pages Web en temps réel, et la création de sites et de pages Web spécialement pour PDAs.

2.3.1 Adaptation des sites et des pages Web en temps réel

Ces solutions s'attellent à convertir automatiquement les pages Web et représenter les sites Web dans le format approprié.

WebTwig & Power Browser

WebTwig et Power Browser (que nous verrons dans la suite de ce mémoire) sont deux exemples de systèmes qui permet à l'internaute de visiter un site Web sans avoir à télécharger les pages de celui-ci. En effet, ils effectuent un certain nombre de traitements, sur un proxy dédié, de manière à rendre les pages et le site Web optimisés pour le PDA.

Il s'agit d'un véritable challenge: les sites Web ne sont pas toujours très bien structurés; et les pages Web incluent de nombreux types objets comme des `tables`, des `frames`, des balises de type ALT (affichant le texte descriptif d'une image lorsque le curseur de la souris passe dessus)... mais également des images, des vidéos, des documents de formats différents, voire même des animations.

W3C

Le Web Consortium (W3C) est en train de développer un plan qui permettra à une page Web d'être créée une seule fois pour toute, et d'être lue sur n'importe quelle plate-forme: PDA, WebTV, téléphone sans fil,etc. L'idée est qu'au moment du transfert des pages Web une négociation soit entamée entre le serveur Web et le système client. Ceci permetra au serveur de connaître les particularités de la machine cliente et de lui fournir la page demandée dans la forme appropriée.

2.3.2 Sites spécialement conçus pour les PDAs

Les solutions envisagées ici ont la particularité de ne pas avoir accès au WWW mais de créer un Web parallèle. Il s'agit en fait de créer des sites spécialement dédiés aux mobiles. Elles contournent ainsi les difficultés d'accession au véritable Web.

Ces approches pour les plus connues sont le WML (Wireless Mark-up Language) pour le protocole WAP (Wireless Application Protocol) et le HDML (Handheld Device Mark-up Language).

WAP & HDML

Brièvement, le protocole WAP développé par le WAP Forum [2] a permis aux téléphones mobiles d'accéder à une riche source d'informations. Il peut naturellement être utilisé avec un PDA. Il a été largement utilisé en Europe et aux États-Unis pour fournir de nouveaux services aux mobiles. Le WAP a aussi été critiqué par beaucoup de personnes dont Nielsen par exemple, qui l'a nommé «Wrong Approach to Portability» ([Nielsen, 1999a]). On peut se demander si l'échec n'est pas dû à une mauvaise approche en CHI, n'ayant pas su mettre l'utilisateur au centre du développement ([Buchanan et al., 2001]).

[2] voir `http://www.wapforum.org`

Quant au HDML, il a été développé avant le WML et a pour but de définir un langage de balise de type HTML propre aux mobiles.

Certains voient dans les solutions de type WML un risque de créer un nouveau Web et d'empêcher les utilisateurs d'avoir accès au véritable Web, nettement plus riche. Nous sommes donc naturellement attirés vers des solutions de type transformation en temps réel.

Il n'est toutefois pas impossible d'imaginer des solutions mixtes: le projet *GreenstoneToAvantGo* en est un exemple.

2.4 La Recherche d'informations sur l'Internet: un problème universel

Le problème de l'accès à l'information d'Internet n'est pas nouveau et n'est pas spécifique aux PDAs. C'est pour cette raison que nous examinerons aussi ce problème d'un point de vue global.

2.4.1 Faciliter la recherche sur l'Internet

Le processus d'accès à l'Internet en vue d'y retirer de l'information suppose un certain nombre d'actions: introduction de données, navigation, recherche par mots clés,...

Les questions de design pour la mise en place d'un système de navigation efficace doivent avoir pour but de simplifier la découverte de l'information recherchée.

Nous entendons par système de navigation efficace, l'ensemble des technologies de l'Internet permettant de poursuivre l'objectif décrit ci-dessus. Par exemple, une meilleure structuration du système des liens hypertextes ou une meilleure qualité des systèmes d'Informational Retrieval (IR).

Information Retrieval & Résumé automatique

L'Information Retrieval (IR) est une composante importante pour la recherche d'informations sur l'Internet: il permet de retirer de l'information de grandes collections de documents par le biais de requêtes sous forme de mots clés. En général, afin de présenter les résulats de la recherche, un résumé

par document retrouvé est fourni. Cela permet à l'utilisateur de prendre assez vite connaissance de l'information qu'il est susceptible de retrouver dans tel ou tel document.

Ces techniques de résumé sont utilisées abondamment par les moteurs de recherche afin de présenter de manière concise les résultats d'une requête.

Structuration des liens hypertextes

Les études portant sur de meilleurs systèmes de liens hypertextes, qui sont au coeur des processus de design de sites Web sont très intéressantes. Un bon système de liens hypertextes facilite la découverte de l'information car il favorise la compréhension que peut avoir l'internaute de la structure du site Web.

2.4.2 Recherche d'informations sur l'Internet via les PDAs

Bien entendu, les problèmes d'accès à l'information ne sont pas propres aux PDAs. Mais ils sont néammoins décuplés par ces derniers. En effet, les PDAs connaissent un grand nombre de contraintes, dont les principales sont les suivantes:

- les PDAs n'ont pas une large bande passante (i.e. le temps de charger une page Web, par exemple, peut prendre quelques dizaines de secondes);
- l'espace d'affichage est réduit (i.e. les utilisateurs sont obligés d'interagir fréquemment avec leur écran).

Afin de pallier ces difficultés il faut principalement:

- minimiser la taille des données à transférer sur le réseau (e.g. minimiser le nombre et la taille des pages Web) tout en assurant une recherche efficace;
- simplifier la structure des sites Web, et des pages Web.

Pour ces raisons toute évolution des technologies de l'Internet améliorera considérablement l'efficacité des PDAs sur l'Internet.

2.5 Introduction de données

Dans l'Internet l'introduction de certaines données est inévitable, ne serait-ce que pour les recherches qui exigent la saisie de mots clés.

Il est évident qu'une solution d'accès à l'Internet qui se veut efficiente, se doit d'offrir des facilités pour ce type d'interaction. Avec la navigation l'introduction de données est le second problème à résoudre pour les PDAs.

Prenons un exemple, et voyons ce qu'il adviendrait de l'utilisateur qui souhaite faire une recherche sur le mot `anticonstitutionnellement`... En français les mots de cette taille sont heureusement rares, mais certaines langues y sont souvent confrontées. Même s'ils sont de taille réduite, l'introduction répétitive des mêmes termes peut vite s'avérer pénible: introduction du/des mot(s) sur le moteur de recherche, sélection d'un site Web, recherche du/des même(s) mot(s) à l'intérieur de la page, et ainsi de suite jusqu'à l'achèvement de la tâche.

2.5.1 Proposition automatique de mots

Les chercheurs ont mis en oeuvre des solutions très intéressantes et très efficaces pour pallier ces difficultés. On retrouve notamment comme technique largement utilisée la proposition automatique de mots: l'utilisateur commence à introduire un mot, et à l'aide d'un dictionnaire, le PDA propose dans une liste défilante les mots qui concordent avec le début du mot introduit. Ainsi, au moment où l'utilisateur écrit `anticons` le PDA offre déjà une série de mots commençant par `anticons` (c'est-à-dire: `anticonstitutionnel(le)`, et `anticonstitutionnellement`), ne restant à l'utilisateur qu'à sélectionner le mot souhaité. En faisant une petite addition on se rend compte que le nombre d'interactions avec l'écran est passé de 25 à 10 pour la saisie du mot le plus long de la langue française.

Évidemment, comme nous le verrons dans la suite, les techniques pour la proposition automatique ne s'arrête pas à l'utilisation d'un dictionnaire usuel. Certaines tiennent compte des mots présents sur la page examinée, ou bien même du contexte de la recherche (par exemple la médecine, l'astronomie, ou autres). En effet, imaginez un instant un jeune étudiant en Chimie faisant une recherche sur la vitamine B 1 qui est tout simplement un `chlorure d'aminométhylpyrimidinylhydroxyéthylméthythiazolium`. On est bien obligé d'imaginer des raccourcis[3].

[3]Les chimistes sont sur ce point imbattables. Le plus long de ces monstres serait le

Les possibilités pour l'introduction d'input sont nombreuses, outre celles proposées ci-dessus; certains travaille sur les techniques d'input de données par la voix.

2.6 Conclusions

Ajourd'hui, tous les éléments techniques sont présents pour favoriser l'émergence des PDAs pour l'accès à l'Internet: infrastructure de téléphonie mobile et puissance des PDAs. Or il reste encore des efforts à fournir pour que le plein potentiel d'utilisation soit atteint. En effet, la recherche d'informations sur le WWW est principalement déterminée par l'aspect humain. Donc, afin de construire un système efficace il faut mettre l'utilisateur au centre du processus de conception. Nous devons donc identifier quels sont les éléments qui posent des problèmes aux utilisateurs pour la découverte d'informations, et les solutionner. Ce sont ces problèmes que nous verrons en détail dans les Chapitres 3 et 4.

nom développé de la *protéine synthétase* $AC_{1289}H_{2051}N_{343}O_{375}S_8$ qui s'écrit avec 1 913 lettres!

Chapitre 3

Recherche sur le Web

L'objectif de ce chapitre est d'identifier les comportements qu'ont les utilisateurs pour trouver de l'information sur l'Internet. Nous examinerons dans la Section 3.2, à travers une expérimentation, comment les utilisateurs s'y prennent pour effectuer un certain nombre de tâches sur le World Wide Web lorsqu'ils disposent d'un petit écran. Nous dégagerons des principes de design pour un système de navigation efficace, spécifique aux PDAs. Nous verrons à la Section 3.3 que le design des systèmes de liens hypertextes n'est pas un problème trivial, et que de nombreux enseignements sont de mise dans le contexte des PDAs. Et finalement, nous présenterons dans la Section 3.4 trois exemples d'implémentations mettant en oeuvre les principes mis en lumière dans ce chapitre: WebTwig, Power Browser et Knowledge Agent Bases.

3.1 Introduction

Accéder à l'information implique un large éventail d'actions allant de l'utilisation de mots clés pour une recherche très précise à la navigation plus ou moins aléatoire.

Par exemple, l'utilisateur a le choix de se rendre directement sur le site Web dont il connaît l'adresse, ou bien d'utiliser un moteur de recherche de type google (`www.google.com`). S'il opte pour la deuxième méthode, entre l'instant où il saisit les mots clés propres à sa recherche et le moment où il trouve l'information, il aura navigué à l'intérieur de plusieurs pages Web — y compris la/les page(s) de résultats sur google —, utilisant un grand nombre de fois la barre de défilement afin de juger de la pertinence des pages.

Il est commun pour une session que l'utilisateur commence avec une recherche partiellement définie qui est ensuite raffinée lors de la phase de navigation, lorsqu'il trouve plus d'informations sur le sujet. La navigation peut donc aider à formuler une requête plus spécifique.

3.2 Améliorer l'Accès à l'Internet des PDAs

L'expérience menée ([Jones et al., 1999a]) que nous allons présenter dans cette section, avait pour but:

1. de quantifier l'impact de l'utilisation d'un petit écran pour accéder à des sites non adaptés;
2. et d'aboutir à des résultats qualitatifs sur la manière dont l'affichage affecte l'approche de l'internaute à la recherche d'informations.

Nous jetterons les premières bases sur la formalisation des processus du comportement de l'utilisateur menant à la création de sytèmes de navigation efficients.

3.2.1 Expérimentation

L'expérience a été effectuée avec la collaboration de vingt informaticiens volontaires agés de 18 à 45 ans. Ils devaient s'acquitter de deux tâches simples sur le site de Reuters[1]. La première était une recherche d'éléments précis:

1. Trouver le prix de l'action d'une compagnie donnée;
2. Evaluer les performances de cette compagnie.

La seconde était une recherche moins dirigée:

1. Trouver le continent avec le plus grand nombre de congés en Mai 1998;
2. Sélectionner le congé le plus intrigant.

Afin d'évaluer les effets de l'écran uniquement, les membres des deux groupes disposaient d'un clavier et d'une souris. On évite ainsi les problèmes dus aux techniques d'input déjà abordés à la Section 2.5.

[1] Compagnie fournissant de l'information financière mondialement, voir `http://www.reuters.com`

Concrètement, l'expérience a été menée sur des PCs dotés d'un navigateur de type Netscape. Afin de simuler la vision reçue par un PDA, la fenêtre du navigateur a été reduite de manière à ne montrer que 15 lignes sur un écran configuré avec une résolution de 640×480 pixels. Pour résoudre chacune des tâches, les membres des deux groupes disposaient chacun de 15 minutes.

Toutes les actions ont été automatiquement enregistrées dans des fichiers journaux pour produire une série de statistiques sur les performances des utilisateurs, sur leurs actions, etc. A la fin des tests, ils ont répondu à un questionnaire permettant de juger leur perception du système.

3.2.2 Résultats et Conclusions

Les résultats ont montré que les utilisateurs de l'écran normal ont répondu deux fois mieux aux exigences des tests. Une analyse des fichiers journaux sur base des statistiques et de la théorie des probabilités — tenant compte du nombre de liens suivis, du nombre de retours sur les pages précédentes, du nombre de défilements sur la hauteur et la largeur des pages Web, ainsi que du nombre de fois qu'un bouton a été pressé — nous permet d'affiner notre vision sur la façon dont les utilisateurs procédent. Ces informations sont indispensables pour tirer des leçons de design. Voyons ces enseignements ...

Afin de concevoir des sites Web facilement accessibles pour PDAs un certain nombre de recommandations ont été proposées. Tout d'abord il faut fournir un « accès direct » à l'information du site. Cela peut être fait par le biais d'outils de recherche par mots clés. Cette étude met en lumière que les «personnes du petit écran» utilisent systématiquement ce genre d'astuce pour trouver l'information, minimisant de la sorte la navigation aléatoire via les liens hypertextes. Ensuite, il faut diminuer les longueurs des pages car les défilements perturbent les utilisateurs. On peut réduire ces défilements de trois manières:

1. Placer des raccourcis de navigation (barres de menu, etc.) près du haut de la page, à une place fixe;
2. Placer des informations clés en haut de la page;
3. Diminuer la quantité d'informations contenues dans une page en la rendant plus concise et moins verbeuse. [Morkes and Nielsen, 1997] adopte également cette règle arguant que les internautes optent pour une technique visant à scanner le contenu des pages plutôt qu'une lecture assidue.

Les propostions faites ci-dessus nous guident vers les caractéristiques que doit offrir un site approprié aux PDAs. Mais ces recommandations prennent également tout leur sens pour les systèmes qui formattent en temps réel des pages Webs à la demande (voir Section 2.2).

[Jones et al., 1999a] ont souligné l'intérêt qu'avait le procédé développé par [Theng et al., 1996a]. Il s'agit de la mise en place d'un système créant une structure de liens hypertextes dédiée à la navigation par but: on propose une liste d'objectifs que l'utilisateur peut accomplir sur ce site. Il faut donc identifier les raisons pour lesquelles un internaute visite ce site et lui proposer une structure de liens qui lui facilite le travail.

Il est clair que le travail fourni et présenté dans ce chapitre est intéressant pour les chercheurs. Il leur permet d'approcher les problèmes liés à l'accès d'Internet par le biais de PDAs en référence à des mesures quantitatives et qualitatives précises.

3.3 Design de sytèmes hypertextes

Les sytèmes hypertextes sont difficiles à concevoir: ils se composent de réseaux riches en noeuds (les pages Web) et en liens (les liens hypertextes). De plus, de nombreuses structures sont possibles, ainsi que maintes façons de les produire. Il s'agit d'une nouvelle discipline et il existe peu d'outils et peu de procédures pour aider les designers à construire un système hypertexte bien structuré. Dès lors, la plupart des sites Web sont mal conçus ([Theng et al., 1996a]).

Il n'est pas absolument certain que les difficultés des liens hypertextes soient uniquement dues à un problème de design. Il se pourrait également qu'il y ait une cause psychologique.

3.3.1 Lost in the hyperspace (LIH)

La richesse qu'offrent les liens hypertextes s'avére, en pratique, rapidement être source de problèmes. L'utilisateur perd son chemin; n'est pas capable de trouver l'information qu'il cherche malgré qu'il sait qu'elle existe; n'est pas capable de se rappeler de l'essentiel de l'information vue.

Ce phénomène nommé «lost in the hyperspace» (LIH) décrit le problème que les internautes peuvent rencontrer lorsqu'ils utilisent des liens hypertextes,

c'est-à-dire être réellement perdus et pas simplement désorientés.

Selon [Theng et al., 1996b] ces problèmes ne sont pas liés uniquement à la perception de l'utilisateur ni même à un problème psychologique. Mais il s'agit là d'un problème impossible à résoudre :«*However, the repeated emphasis on user problems (ironically for such a promising technology!) seems suspicious. It seems to us, rather that anything would get lost in current hypertext system (By "anything" we mean any device capable of algorithmic thought). If so, then the problems is not based in user perception nor in user psychology.*».

Il s'avère donc indispensable de repenser le design des sites Web afin de réduire au maximum les effets du LIH. Comme nous l'avons déjà mentionné dans la section précédente (et qui a été souligné par Jones et al.) un système de navigation par but peut être très intéressante.

3.3.2 Identification des causes du LIH

[Theng et al., 1996b] suppose deux raisons au phénomène du LIH. La première est un problème de design: s'il est de mauvaise qualité, il posera des problèmes psychologiques à l'internaute. Et la seconde est un problème lié aux utilisateurs: il se pourrait qu'ils soient incapables d'exploiter l'écran et la structure complexe de l'information.

Les hypothèses sur les utilisateurs sont les suivantes ([Theng et al., 1996b]):

- Les utilisateurs ont un modèle conceptuel faux ou incomplet de la manière dont l'information est structurée et liée à l'intérieur du système hypertexte.
- Les utilisateurs font face à un problème de "embedded digression" où ils se laissent distraire par une foule d'informations intéressantes.
- Les utilisateurs ont un manque d'expérience dans l'utilisation de l'hypertexte. Cela entraîne qu'il leur est difficile de comprendre la sémantique d'une page et de résumer ce qui a été vu.
- Les utilisateurs n'emploient pas tous leurs sens. Il devrait être possible d'utiliser des données sonores par exemple. Cette idée est d'ailleurs explorée en Réalité Virtuelle.

3.3.3 Conclusions

En conclusion, *il est important de bien connaître les utilisateurs et leurs besoins, et il est indispensable de se faire une idée précise du comportement qu'ont les internautes ainsi que des actions qu'ils peuvent accomplir.*

Cette vision a inspiré les deux systèmes de navigation que nous verrons dans la section suivante: le WebtWig et le Power Browser.

3.4 Sytèmes de navigation pour PDAs

Comme nous l'avons vu, un système de navigation efficace doit supporter un accès direct à l'information. Les systèmes qui vont vous être présentés utilisent l'approche de transformation de pages Web par l'intermédiaire d'un proxy (voir Figure 3.1, [Buyukkokten et al., 2000a]).

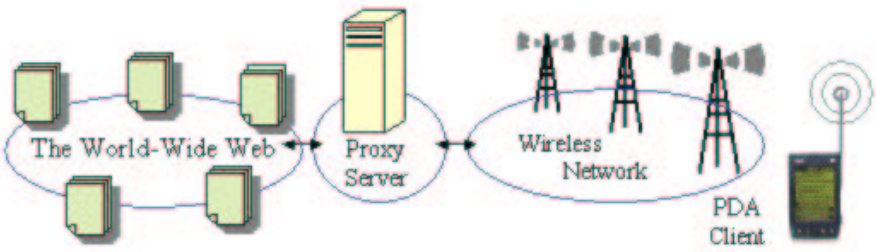

FIG. 3.1 – *Le proxy est l'intermédiaire entre les PDAs et le WWW: il effectue tous les traitements avant de retourner les résultats sur les PDAs.*

3.4.1 WebTwig

Le sytème WebTwig est développée par [Jones et al., 1999b].

Naviguer

WebTwig permet de visualiser un site Web en donnant aux utilisateurs un haut niveau d'appréciation de l'information contenue. Afin de se donner une idée du résulat observons les Figures 3.2 et 3.3. La seconde figure nous montre comment le WebTwig nous laisse voir le site de la première figure: il

représente le site par l'intermédiaire d'un arbre. L'utilisateur peut «visiter» le site en cliquant sur un noeud de l'arbre. Ce qui a pour conséquence d'ajouter un sous-arbre à l'arbre initial, et donc d'affiner la vision du site:l'arbre s'étend. Le système est similaire aux explorateurs de fichiers sur les systèmes d'exploitation traditionnels.

Cela permet donc à l'utilisateur d'identifier les parties intéressantes du site sans avoir à télécharger toutes les pages HTML du site, souvent trop complexes pour les PDAs. Il s'agit également d'un accès direct à l'information.

FIG. 3.2 – *Site Web qui va être transformé par Webtwig.*

3.4.2 Power Browser

Le «Power Browser» ([Buyukkokten et al., 2000b]) a été développé à l'Université de Standford.

Trouver l'information

L'analyse des personnes faisant des recherches sur l'Internet peut être schématisée par la Figure 3.4 ([Buyukkokten et al., 2000a]).

La Figure 3.4(1a) schématise une recherche typique sur l'internet: l'utilisateur consulte un moteur de recherche pour une recherche global. Une fois qu'il

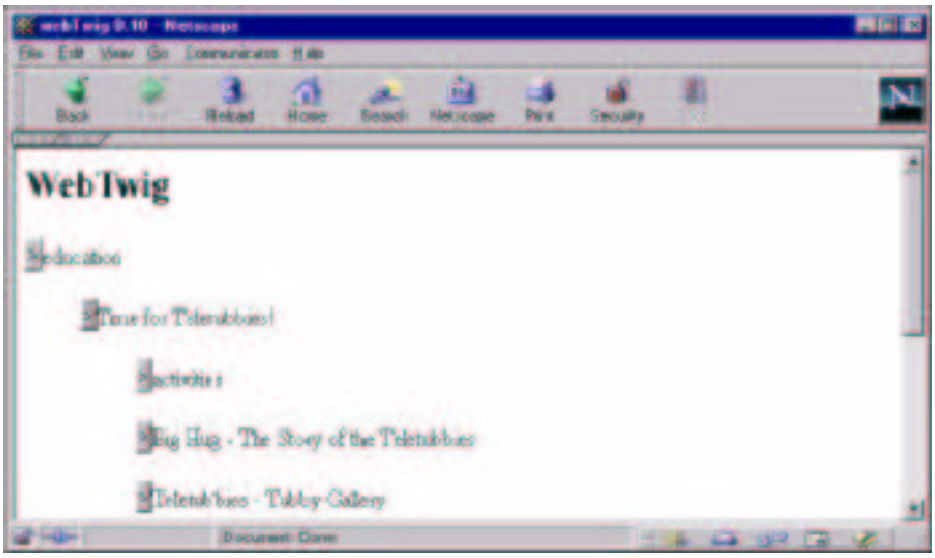

FIG. 3.3 – *Vue du site Web proposée par WebTwig.*

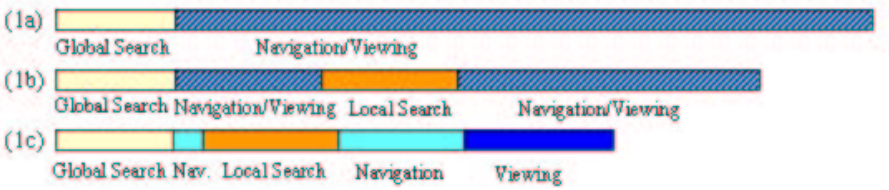

FIG. 3.4 – *Méthodes de recherche sur le Web. L'axe horizontal représente le temps.*

reçoit les résultats, il entre dans une phase de navigation et de visualisation, jusqu'à ce que la page dont il a besoin soit localisée.

Sur la Figure 3.4(1b) on voit une amélioration: certains sites Web offrent un moteur de recherche. Après la recherche global, l'utilisateur visualise les sites web référencés et recommence une nouvelle recherche, à l'aide de mots clés, à l'intérieur du site. Ceci a pour effet de réduire le temps de recherche.

Sur la Figure 3.4(1c) on aperçoit le système proposé par le Power Browser: il permet de faire une recherche sur un site, même pour les sites n'offrant pas de moteur de recherche.

Principe Le Power Browser indexe toutes les pages des sites Web par des mots clés.

Naviguer

A l'instar du WebTwig le système de navigation supporte la structure arborescente des sites Web (Figure 3.5). La différence réside essentiellement dans le fait que le Power Browser fournit la structure de l'arbre en considérant la page courante comme la racine de l'arbre, alors que le WebTwig montrait l'ensemble du site.

Ainsi l'internaute peut directement accéder à l'information en cliquant sur le lien désiré: il doit faire un geste de droite à gauche avec son stylus sur le lien qu'il veut suivre. L'écart des lignes verticales représente le niveau de l'arbre.

Introduction de mots clés

Si l'utilisateur souhaite faire une recherche dans un site Web, un message est envoyé au proxy qui charge en mémoire l'index correspondant au site. Il met à jour l'index lorsque l'utilisateur visite des pages qui ne l'ont pas été auparavant. Si l'index n'est pas disponible (i.e. le site n'a jamais été visité), il est créé sur le champ.

Proposition automatique de mots

Le Power Browser offre un système de proposition automatique de mots spécifique au site Web. Le système informe en temps réel combien de pages

FIG. 3.5 – *Structure arborescente du Power Browser.*

concordent.

Lorsque l'utilisateur entre les premiers caractères, une requête est envoyée au proxy qui retire tous les mots du site Web qui commencent par ces caractères. Le résultat est affiché sur le PDA dans une liste déroulante. Si le mot s'y trouve il peut le selectionner. Chaque fois que l'utilisateur introduit plus de caractères la liste est mise à jour.

3.4.3 Knowledge Agent Bases (KABs)

Le système, Knowledge Agent Bases (KABs), est une nouvelle approche qui introduit dans le processus de recherche un «agent technology». Cet agent est situé entre l'utilisateur et le moteur de recherche [Aridor et al., 2001]).

Trouver l'information en mode non connecté

Le principe général est de favoriser la recherche d'informations en mode déconnecté:

1. l'utilisateur formule une requête sur le système KAB sans être connecté à l'Internet;

2. lorsqu'il se connecte à l'Internet, la requête est envoyée au proxy qui effectue la récupération de l'information sur le Web;
3. le proxy renvoie les résultats sur le PDA;
4. finalement, lorsque l'utilisateur se déconnecte, il peut commencer sa recherche sur son PDA.

Pour ce faire, la représentation de l'information est basée sur les «knowledge-agent bases». Ils encapsulent l'information nécessaire pour assister l'utilisateur dans son processus de découverte de l'information sur le PDA.

L'idée est d'autoriser l'utilisateur à prédéfinir un sujet qui l'intéresse, et ensuite, à capturer une toute petite mais représentative portion du Web pour ce sujet, et de stocker ces informations sur son PDA.

Les informations stockées sont:

1. un lexique du sujet;
2. une centaine de pages Web pertinentes pour le domaine prédéfini (*core pages*), pour lesquelles l'entièreté des textes sont disponibles;
3. de l'ordre de mille URLs issues des liens hypertextes des *core pages*;
4. et finalement, un index des mots clés construit à partir des *core pages* et des liens hypertextes. Cet index permet de supporter la recherche par mots clés dans le KAB.

Les KABs sont construits automatiquement: les utilisateurs n'ont qu'à fournir 4 à 6 requêtes qui définiront le sujet, et éventuellement un ensemble d'URLs qu'ils considèrent représentatifs du sujet.

Recherche par mots clés et proposition automatique de mots

KABs supporte la recherche par mots clés et la proposition automatique de mots basées sur l'index construit préalablement. Les résultats d'une recherche sont présentés par ordre de pertinence en ne montrant que les titres des *core pages* et les liens hypertextes (un peu à l'image des moteurs de recherche conventionels). Les utilisateurs peuvent sélectionner un lien et obtenir plus d'informations, ou voir l'ensemble de la page Web s'il s'agit d'une *core page*.

3.4.4 Evaluation

WebTwig & Power Brower

Ce sont deux systèmes très intéressants car comme nous l'avons vu, il est possible de visiter entièrement un site Web (et faire des recherches par mots clés en local [dans le cas du Power Browser]) sans devoir visualiser la moindre page Web du site.

Un des challenges de ces approches est le choix d'une bonne description pour chaque noeud de l'arbre (nous verrons d'ailleurs dans le Chapitre 4 comment répondre à ce genre de problème). Le Power Browser emploie des méthodes heurisitiques, par exemple, en utilisant les titres des pages pointées par les liens hypertextes. Cette technique donne de bons résultats, mais malheureusement, elle réduit considérablement les performences du système.

Les qualités des systèmes dépendent beaucoup de la cohérence du site Web. Si un site ne présente pas suffisamment de structure, WebTwig sera incapable d'en extraire une représentation sous forme d'arbre.

Un autre souci découle de la définition d'un site Web: plusieurs groupes peuvent partager un même site Web et un site Web peut être déployé sur plusieurs serveurs. La qualité de la recherche par mots clés, en local, dépend aussi de ces questions.

KABs

La recherche d'informations en mode déconnecté est très avantageuse. Elle réduit les problèmes dus à la bande passante, en évitant d'accéder régulièrement à l'Internet et réduit les coûts de connections.

La solution est intéressante lorsque l'utilisateur a une bonne idée de ce qu'il recherche. Mais, il ne peut pas toujours formuler son désir, et ne connaît pas nécessairement le vocabulaire approprié pour décrire le sujet qui l'intéresse. Et dans certains cas, il ne souhaite pas choisir de mots en particulier, mais plutôt découvrir de l'information en général. La solution reste tout de même intéressante mais dans ce cas on suppose que la personne peut se connecter régulièrement à l'Internet.

3.5 Conclusions

Dans ce chapitre nous avons pu identifier la manière dont les utilisateurs recherchent de l'information sur le World Wide Web par l'intermédiaire d'un PDA. Nous avons aussi vu que la difficulté des liens hypertextes est un problème global, et qu'un système de navigation efficace pour PDA passe par une amélioration du design des sites Web. D'ailleurs, certaines propositions formulées, avant les questions suscitées par les PDAs, ont été prises en compte pour la mise en place des systèmes que nous avons vus au travers du WebTwig et du Power Browser.

Comme nous le verrons plus tard, la raison d'être des bibliothèques digitales est de présenter l'information dans un système de liens hypertextes cohérent et bien structuré. De plus, une bibliothèque digitale offre par nature un système IR de qualité permettant une recherche par mots clés dans une grande collection de documents.

C'est une des raisons qui a motivé le développement de l'application GreenstoneToAvantgo. En effet, en fournissant un accès limité aux bibliothèques digitales (construites par le software Greenstone), notamment en simplifiant les pages HTML, GreenstoneToAvantgo réunit d'un coup deux éléments essentiels à la découverte de l'informations: un bon système de liens hypertextes (facilitant la navigation), et un bon système de recherche par mots clés.

A l'instar du «Knowledge Agent Bases» (KABs), GreenstoneToAvantgo fonctionne en mode déconnecté. Il bénéficie donc des avantages qu'offre cette approche mais pas des inconvénients soulignés précédemment. Car GreenstoneToAvantgo stocke une portion de la bibliothèque digitale, et donc également sa structure. Ceci a pour conséquence, que GreenstoneToAvantgo supporte la navigation en mode déconnecté.

GreenstoneToAvantgo fournit également, à la manière des KABs, des possibilités de recherche par mots clés en local. Reste dès lors les questions relatives à la compression, qu'offre GreenstoneToAvantgo, pour le stockage des informations. Nous aborderons ces questions dans la suite de ce travail.

Chapitre 4

Les Résumés automatiques

Le but de ce chapitre est de mettre en lumière les différents avantages que l'on peut tirer des techniques de résumé généré automatiquement pour la mise en place d'un système de navigation efficace. Loin d'avoir la prétention de traiter ce sujet de manière complète, nous offrirons tout au plus une introduction nous permettant de saisir les concepts sous-jacents.

Nous verrons dans la Section 4.2 les différents termes et définitions associés à cette activité. Nous examinerons à la Section 4.3 les bases des systèmes d'Information Retrieval (IR) dont la qualité a un impact direct sur la recherche dans l'Internet. Les procédés de génération automatique de résumés sont indissociables de l'IR, et nous verrons dans la Section 4.4 quelques techniques propres à l'IR, ainsi qu'un algorithme d'extraction de mots clés d'un document. Nous présenterons dans la Section 4.5 le principe général d'une méthode d'Information Extraction (IE) utilisée comme alternative aux techniques IR.

A la Section 4.6, nous examinerons comment les principes vus peuvent être utilisés dans le World Wide Web; notamment par l'exemple d'OCELOT, un prototype qui génére automatiquement des résumés de pages Web. Nous dégagerons également certains enseignements sur les potentiels offerts par le Web. Et finalement, grâce à l'exemple du Power Browser, nous verrons à la Section 4.7 comment améliorer les sytèmes de navigation pour PDAs.

4.1 Introduction

Bien évidemment, le résumé n'est pas une idée nouvelle: pour s'en convaincre il suffit de voir à quel point les titres soigneusement rédigés et placés dans les journaux nous aident à avoir une idée générale du contenu, afin de décider si l'article vaut la peine d'être lu.

La nécessité d'effectuer cette tâche de manière automatique s'est avérée très tôt indispensable parallélement au domaine de l'Information Retrieval (IR). C'est avec Luhn (1955) et Edmunson (1969) qu'ont été posées les premières pierres d'une activité très complexe, pour laquelle les approches et les discussions se sont succédées et pour laquelle aucune solution n'a jamais vraiment satisfait l'ensemble de la profession. Les aspirations sont différentes, certains voient dans cette activité la nécessité de produire un résumé permettant au lecteur d'avoir une idée du contenu du texte, tandis que d'autres plus ambitieux requièrent un nouveau texte reformulé, cohérent et condensant le document d'origine. Chacun de ces objectifs est fondé et a donné de bons résultats dont l'évaluation est souvent propre au contexte de travail dans lequel on les applique. Ainsi, certaines méthodes fonctionnent mieux avec des articles de presse. Cet ensemble de possibilités crée l'enthousiasme auprès d'un très grand nombre de chercheurs.

Afin de donner un contexte, voyons un court historique des références dans ce domaine. Il a été exposé par le professeur de Computational Linguistics Udo Hahn's de l'Université de Freiburg dans la présentation faite à Seatle lors du «Automatic Summarization Workshop» (2000) (cité dans [Barbosa, 2001]):

Early Extraction 1955-1973

- Luhn, 1955, lexical occurence stats
- Edmundson, 1969, positional indicators, lexical cues, cuewords, cue phrases
- Mathais, 1973, cohesion streamlining

Linguistic Approaches 1961-1979

Psychological Approaches 1975-1985

Artificial Intelligence Approaches 1980's

- 1982. Dejong. FRUMP, using scripts

- 1985. SUSY, logic and production rules
- 1988. Reiner and Hahn. TOPIC Frames and semantic networks
- 1989. Rau et al. Hybrid Representations

«Renaissance» 1990's

- Recurrence of statistical techniques
- Hybrid approaches

Comme le lecteur peut s'en rendre compte, il serait vain dans le cadre de ce mémoire d'être plus exhaustif.

Traditionnellement, deux paradigmes se partagent ce monde. Le premier, Information Retrieval (IR) utilise à la base des techniques statistiques, et il n'est pas rare d'entendre dans la littérature parler de méthodes statistiques ou bien «bottom-up». Le second, Information Extraction (nouveau champ en Natural Language Processing [NLP]) utilise des procédés dits «symboliques» ou encore «top-down».

4.2 Termes et définitions

De nombreuses définitions de résumé, dépendant de l'approche préconisée, ont été proposées. Mais afin de donner un cadre et de n'en favoriser aucune nous proposons la définition neutre suivante (Oxford Dictionnary):

> **summary:** a short statement that gives only the main points of something.

On peut dire que le fait de résumer un document (un ensemble de documents ou autre chose) a pour but de produire un autre document dont le dégré de sophistication dépend de la technique utilisée. Un résumé peut être une liste de mots clés indiquant les points essentiels, une liste de phrases pertinentes prises telles quelles dans le texte, ou bien un condensé cohérent avec notamment des reformulations. A ce titre nous présentons ci-dessous les deux grandes distinctions quant aux différentes variations applicables pour un résumé [Edmundson, 1969]:

- *Extract vs Abstract*: un *extract* est une sélection de phrases du document original, tandis qu'un *abstract* est une condensation et une reformulation de l'original.

– *Informative vs Indicative*: un résumé *informative* peut servir de substitut au document original tandis qu'un résumé *indicative* nous donne suffisamment d'informations pour nous permettre de savoir si le document est intéressant;

Bien que *abstract* sous-entende bien plus de choses qu'*extract*, il est fréquemment utilisé de manière générique pour nommer les deux concepts.

Il est bien évident que certaines de ces variations demandent plus ou moins d'efforts. Prenons par exemple la différence entre *abstract* et *extract*. Pour obtenir un *extract* des méthodes de statistiques pures, basées sur la distribution des fréquences des mots ou la position des phrases, peuvent être suffisantes dans certains cas. Nous sommes ici dans la cour de l'Information Retrieval.

Par contre, si le désir est de produire un *abstract*, il est clair qu'on joue plutôt au niveau sémantique du texte et donc qu'un certain niveau de compréhension de sens est nécessaire. C'est ce que l'Information Extraction entreprend.

4.3 Information Retrieval (IR)

Avant de commencer l'étude à proprement dite des systèmes proposés par l'IR, voyons les éléments clés qui sont à sa base, ainsi que le cadre dans lequel il évolue.

4.3.1 Tâches Principales

Afin de maintenir des données dans un système informatique on peut au moins compter sur [Barbosa, 2001]:

1. un système de gestion de base de données (database management systems [DBMS]): l'information est organisée dans des structures de données, tels que des `tables`.
2. Ou un système IR: des textes sont maintenus dans un langage naturel à l'intérieur de grandes collections de documents.

Dans le premier système, un ensemble de mécanismes permet de localiser et de retirer rapidement l'information voulue, lui permettant de supporter l'accès direct. Dans le second par contre il n'existe pas de système d'accès direct. Au contraire il faut plutôt faire face à des problèmes d'imprécision:

le système a une connaissance imprécise du contenu. Et à des problèmes d'approximation: l'utilisateur ne peut pas décrire l'information dont il a précisément besoin [Fuhr, 2001]. Il existe donc des méthodes ad hoc. Typiquement une requête en IR est composée d'opérateurs booléens AND-NOT-OR combinés à des mots clés grâce auxquels le système doit récupérer un ensemble de documents concordants au maximum avec cette requête.

Selon [Barbosa, 2001] on peut dès lors voir le système du point de vue fonctionnel comme:

- un ensemble de documents;
- un ensemble de requêtes;
- et un ensemble de mécanismes qui permettent de déterminer quelle est l'information concordante avec la requête;

Ce qui nous amène à identifier les trois tâches principales qui doivent être remplies:

1. indexer tous les documents par des mots clés (discriminants) en pondérant leur poids relativement au document associé. De cette manière chaque document sera associé à un vecteur de mots clès le caractèrisant (c'est également la représentation interne du document);
2. retranscrire la requête de l'utilisateur dans le language de représentation interne du système (en utilisant des opérateurs booléens par exemple);
3. et finalement déterminer l'ensemble des documents qui a la plus grande similitude avec la requête de l'utilisateur — en utilisant les index précédemment construits.

Habituellement, la première requête de l'utilisateur sera soit trop générale soit trop spécifique. Ceci aura pour conséquence de produire un résultat ne le satisfaisant pas. C'est est dû au fait que l'utilisateur ne connait pas les mots clés utilisés par le système pour indexer les documents. Ce qui le conduit généralement dans une phase de processus interactif grâce auquel il commence à raffiner sa requête. Il existe des systèmes IRs plus évolués qui font des suggestions à l'utilisateur pour l'aider à choisir les futurs mots clés.

4.3.2 Qualité du Système IR

Afin de juger de la qualité du système on regarde notamment [Fuhr, 2001]:

- l'*efficience*: les ressources systèmes utilisées pour effectuer une tâche

(e.g. temps CPU);
- et l'*efficacité*: le degré de support offert à l'utilisateur pour qu'il puisse résoudre son problème:

$$efficacité = \frac{qualité\,des\,résultats}{quantité\,d'effort\,de\,l'utilisateur}$$

En général, c'est cet indicateur qui définit la performance du système.

La définition de *pertinence* d'un résultat est [Fuhr, 2001]:

- en général *objective*: jugement neutre (d'un expert du domaine) de la relation entre la demande exprimée par l'utilisateur et les documents fournis;
- et parfois, *subjective*: utilité que l'utilisateur retire des résultats.

Ainsi que les deux indicateurs suivants [Fuhr, 2001]:

1. *recall*: la proportion de documents pertinents renvoyés, sur l'ensemble des documents pertinents de la collection satisfaisant à la requête:

$$recall = \frac{\sharp\, documents\, pertinents\, retirés\, de\, la\, collection}{\sharp\, documents\, pertinents\, dans\, la\, collection}$$

2. *precision*: (paramètre directement observable) est la proportion des documents pertinents sur l'ensemble des documents renvoyés (qui ne sont pas nécessairement tous pertinents):

$$precision = \frac{\sharp\, documents\, pertinents\, retirés}{\sharp\, documents\, retirés}$$

Pour expliquer les deux termes *recall* et *precision* nous allons prendre un exemple. Supposons qu'une personne soit intéressée par l'acquisition d'un «voilier de course». Pour commencer sa recherche, elle introduira dans le système trois mots clés (requête qui sera transformée dans une représentation propre au système e.g. «achat» AND «voilier» AND «course»). Supposons qu'il existe 10 documents qui soient pertinents pour l'acquisition d'un bateau de ce type dans le système. Imaginons que le résultat donne 6 documents — dont 4 font partie des 10 documents pertinents— et 2 autres parlent de voilier, de course, d'achat sans parler d'achat **de** voiliers **de** courses.

Documents retirés	Documents pertinents	Recall	Precision
10	4	0.4	0.66
15	8	0.8	0.53
20	10	1	0.5

TAB. 4.1 – *Exemple de recall et de precision dans un système IR, sachant que le nombre de documents pertinents dans le système est de 10.*

En effectuant le calcul du *recall* on obtient un taux de 0.4 (c'est-à-dire les 4 sur les 10 documents pertinents du système) et une *precision* de 0.66 (c'est-à-dire 4 documents valables sur les 6 renvoyés).

En règle générale, plus on augmente le nombre de documents renvoyés, plus on améliore le *recall*, mais plus on diminue la *precision*. Par exemple supposons qu'on demande au système de renvoyer 15 documents, et disons que le nombre de documents pertinents passe de 4 à 8. On augmente le *recall* qui passe de 0.4 à 0.8 et on diminue la *precision* en la faisant passer de 0.66 à 0.53. Le phénomène inverse s'observe également. Ces deux paramètres sont compris dans l'intervalle [0.0;1.0] avec leur optimum en 1. Comme le montre le Tableau 4.1 ils sont inversement proportionnels[1] [Barbosa, 2001]. Il n'est pas évident de calculer le *recall* (car pas directement observable), et à ce titre plusieurs solutions existent.

4.4 Techniques de Résumé en IR

L'intérêt des résumés dans ce domaine est maintenant clair, car en voulant augmenter le *recall* et donc le nombre de documents pertinents, il faut augmenter le nombre de documents renvoyés et donc diminuer la *precision*. Ce qui a pour conséquence inévitable de rendre la recherche parmi les résultats beaucoup plus confuse. La solution n'est autre que pour chaque document présenté, un résumé y soit associé.

[1] Il s'agit d'une observation empirique et non pas d'une relation strictement mathématique.

4.4.1 Anciennes Techniques

Méthodes de fréquence des mots

C'est Luhn en 1958 qui a le premier développé cette idée en prenant en compte la distribution des mots, basée sur l'intuition que les mots les plus fréquents (excepté les mots courants de la langue) sont les concepts les plus importants du texte. Il suggéra donc d'estimer l'importance des phrases d'un document en fonction des fréquences de mots les plus élévées et pour lesquels il y a une forte proximité (cité dans [Hovy and Lin, 1997]).

Les trois implémentations suivantes — qui s'ajoutent à la méthode standard des mots clés (Luhn)— ont été présentées par Edmundson (1969).

Méthode du «Lexical Cue»

Basée sur des informations linguistiques, l'identification de *cue words* utilise des marqueurs méta-linguistiques (par exemple, «En conclusion», «Cet article décrit...», «impossible») pour séléctionner les phrases importantes. Cette technique utilise une liste de *cue words* générée statistiquement grâce à des résumés de référence produits par l'homme (cité dans [Hovy and Lin, 1997]).

Méthode de location

La méthode de location se base sur l'intuition suivante: le début et la fin des textes ou des paragraphes, les phrases mises en gras contiennent les phrases importantes (cité dans [Hovy and Lin, 1997]).

Méthode des titres

Ici le poids des phrases est une fonction de la somme des termes apparaissant dans les titres et les sous-titres du texte (cité dans [Hovy and Lin, 1997]).

Evaluation

Bien que ces approches aient des qualités, elles dépendent pour beaucoup de la forme et du style d'écriture. La stratégie de prendre le premier paragraphe

par exemple ne fonctionne correctement que dans les journaux et les magazines (et pas toujours). Dans l'ensemble l'emploi des trois méthodes ensemble donne de meilleurs résultats. Tandis que la technique de Luhn employée seule n'est pas très efficace.

4.4.2 Extraction de mots clés: la méthode TF/IDF

Afin de pallier les lacunes des systèmes présentés ci-dessus, des méthodes plus complexes ont été élaborées. Nous n'en présenterons qu'une seule : TF/IDF (Salton et Buckley [1990,8]) (cité dans [Barbosa, 2001]), pour la simple raison qu'elle est fréquemment utilisée pour l'extraction des mots clés — et également utilisée par le «Power Browser» (que nous examinerons plus tard dans ce chapitre).

Calculer le poids des mots d'un texte

Une méthode standard d'extraction de mots clés pour résumer (*extract* et *indicatif*) est le système *TF/IDF* (*term-frequency* $*$ *inverse-document-frequency*) (cité dans [Barbosa, 2001]).

En principe, l'extraction de mots clés d'un texte doit considérer l'importance du mot dans ce texte. Une façon de le faire est de considérer le nombre d'occurences d'un mot W dans le document D, par rapport au nombre de fois qu'apparaît W dans une large collection de documents duquel on a retiré D. Intuitivement, un mot est important s'il est présent fréquemment dans un texte mais pas fréquemment dans une large collection. Cette intuition est capturée comme suit:

$$w_{ik} = tf_{ik} \times \log \frac{N}{n}, \qquad où \tag{4.1}$$

$$\begin{aligned} w_{ik} &= poids\ du\ terme\ T_k\ dans\ le\ document\ D_i \\ tf_{ik} &= fréquence\ du\ terme\ T_k\ dans\ le\ document\ D_i \\ N &= nombre\ de\ documents\ dans\ la\ collection \\ n &= nombre\ de\ documents\ dans\ lequel\ T_k\ apparaît\ au\ moins\ une\ fois \end{aligned}$$

Normalisation Les longs documents ont le désavantage d'utiliser un grand nombre de mots différents qui sont plus fréquemment répétés. Un système

juste basé sur tf*idf biaiserait le résultat. C'est pour cela qu'il est convenu de normaliser le résultat de cette façon:

$$w'_{ik} = \frac{w_{ik}}{\sqrt{\sum_k (w_{ik})^2}}, \qquad où \tag{4.2}$$

$$\begin{aligned} w'_{ik} &= \textit{poids normalisé du terme } T_k \textit{ dans le document } D_i \\ w_{ik} &= \textit{poids du terme } T_k \textit{ dans le document } D_i \end{aligned}$$

L'algorithme d'extraction de mots clés d'un texte

L'algorithme associé à l'extraction des mots clés candidats dans un document est le suivant:

1. Identifier individuellement les mots du texte;
2. Utilisation d'un dictionnaire des mots les plus fréquents pour les éliminer de la sélection;
3. Utilisation d'une routine afin de réduire les mots à leur radical;
4. Enlever tous les termes dont le nombre d'occurences n'atteint pas un certain seuil;
5. Calculer le poids des termes par la méthode TF/IDF;
6. Sélectionner les termes en fonction de leur poids qui tiendront lieu de mots clés.

4.4.3 Evaluation des techniques IR en général

Ces techniques semblent être la solution la plus simple à implémenter pour qui veut obtenir rapidement un résumé l'éclairant sur le contenu d'un texte. De plus les utilisateurs sont assez tolérants et apprécient les résultats issus des systèmes IRs. Quoi qu'il en soit, actuellement ces procédés sont les plus concluants pour ce qui est de traiter un texte d'information générale. Comme nous le montrerons les techniques nées de l'Information Extraction donnent de bons résultats lorsqu'elles sont appliquées à des domaines restreints.

4.5 Information Extraction (IE)

Les techniques de l'IR allient des procédés statistiques avec des heuristiques, mais un certain nombre de critiques ont fusé sur le fait que cette approche

ne traitait pas les difficultés du langage naturel[2].

Ces problèmes sont liés à la sémantique et l'approche pure de l'IR a des limitations quand il faut traiter ce genre d'ambiguïtés linguistiques. Cette attitude est la force mais aussi la faiblesse de l'IR. C'est une force car elle leur permet de se libérer de l'idée qu'en se préoccupant de la signification des mots tous les problèmes vont se résoudre comme par magie. C'est une faiblesse également car cela les empêche de raisonner à un niveau plus élevé et de produire des résumés cohérents ([Hovy and Lin, 1997]).

Si l'on veut travailler à un niveau supérieur on devrait exiger l'existence d'une «base de connaissances» («world knowledge») afin de lever les ambiguïtés. La complexité de cette tâche (i.e. produire un *abstract*) est la raison majeure pour laquelle la plupart des systèmes dits symboliques ou bien encore «top-down» s'en sont tenus à des domaines très restreints. Et c'est également pour cela qu'on évite ce genre de solutions lorsqu'on a affaire à des textes de portée générale e.g. le journal.

4.5.1 Principe d'une méthode

La plus récente et plus satisfaisante approche dans ce domaine utilise des formulaires prédéfinis, par un spécialiste du domaine, contenant des vides qui doivent être remplis avec les mots attendus pendant le processus de génération de résumé. Le système reconnaît les types de termes à rechercher et le contexte dans lequel il doit les retrouver. Il suffit donc d'avoir un parseur/analyseur suffisamment puissant pour extraire les pièces d'information appropriées du texte (cite dans [Hovy and Lin, 1997]).

TIPSTER a développé un analyseur de ce type qui traite l'information du

[2]

- *synonymes*: Un même concept peut être exprimé de différentes manières, par exemple *voiture* et *automobile*;
- *homonymes*: Un mot peut avoir plusieurs sens, par exemple un *cousin* (insecte) ou *cousin* (parent);
- *termes anaphoriques*: Un mot peut renvoyer à un autre mot ou à une autre phrase apparue antérieurement;
- *métaphores*: Procédé par lequel on transporte la signification propre d'un mot à une autre signification qui ne lui convient qu'en vertu d'une analogie, d'une comparaison sous-entendue e.g.*la lumière de l'esprit*;
- *métonymies* Procédé par lequel un concept est désigné par un terme désignant un autre concept qui lui est relié par une relation nécessaire e.g. *une fine lame* (escrime);

monde réel sur base du journal. Mais il ne traite que les nouvelles ayant pour sujet le terrorisme. Des sytèmes tels que TIPSTER/MUC, FASTUS (Hobbs 1992), GE_CMU (Jacobs 1990), CIRCUS (Lehnert 1991) et d'autres en sont des représentants intéressants (cités dans [Hovy and Lin, 1997])

4.5.2 Evaluation

On obtient des *abstract informatifs*. Si quelqu'un désire savoir ce que le texte dit et pas seulement ce que l'analyste a défini comme étant intéressant alors cette approche n'est pas adéquate. De plus ce système serait vraiment puissant si un grand nombre de formulaires était créé et que le système puisse déterminer automatiquement lequel il faut appliquer à n'importe quel texte.

4.6 Application au Web

Ce n'est pas surprenant que la plupart des recherches pour accéder à l'information sur le Web se soient concentrées sur l'IR. En effet, on peut voir le World Wide Web comme une gigantesque collection de documents en tous genres.

La plupart des méthodes que nous avons montrées dans la Section 4.4 étaient principalement basées sur la structure homogène des textes. Avec les technologies Web il est possible pour tout un chacun de créer son information et de la rendre disponible sur le WWW. Ce qui a pour conséquence de rendre les documents Web peu cohérents avec une structure, et une langue utilisée n'étant pas clairement définie [Barbosa, 2001]. Bien que le HTML offre pas mal de possibilités pour structurer les pages grâce à des balises (tel que $< \backslash TITLE > \cdots < \backslash TITLE >$), celles-ci ne sont pas obligatoires.

Nous verrons maintenant comment OCELOT aborde ces difficultés.

4.6.1 OCELOT

Ocelot ([Berger and Mittal, 2000]) est un prototype qui fournit des résumés de page Web. Il a la possibilité de générer des textes qui sont plus que des *extracts* du document original. En effet, les résumés peuvent contenir de nouveaux mots. Une autre particularité d'OCELOT est qu'il peut gérer plusieurs


Santé Canada Health Canada

Canada

English | Contactez-nous | Aide | Recherche | Site du Canada
Accueil - SPSP | Centres | Publications | Lignes directrices | Indexe A-Z
Santé - enfants | Santé - adultes | Santé - aînés | Laboratoires | Surveillance

Direction générale de la santé de la population et de la santé publique (DGSPSP)

SCHIRPT

Un grave problème de santé chez les enfants

Au Canada, depuis 10 ans, les blessures chez les enfants sont devenues un grave problème de santé publique. Ce problème n'est pas nouveau, mais ce n'est que récemment que l'on a commencé à en étudier toute la portée. Le taux de mortalité reflète l'ampleur des blessures les plus graves : chaque année, les blessures causent plus de décès chez les jeunes Canadiens et Canadiennes de plus d'un an que toutes les autres causes réunies. Pour chaque décès lié à une blessure, on compte 45 hospitalisations et environ 1 300 visites à l'urgence dans l'ensemble du pays. Ajoutons qu'environ 90 p. 100 de ces blessures sont probablement prévisibles et évitables.

Jusqu'à récemment, on ignorait presque tout des circonstances entourant les blessures chez les enfants. Les taux de décès et d'hospitalisation, bien qu'utiles, ne répondent pas à toutes les questions de ceux qui travaillent à la prévention des blessures. Que faisait l'enfant lorsqu'il a subi la blessure? Où était-il? Que s'est-il passé? Le Système canadien hospitalier d' information et de recherche en prévention des traumatismes (SCHIRPT) est né en 1990 pour répondre à ces questions.

[Section des blessures] [Bureau de la santé génésique et de la santé de l'enfant] [Précédente] [Prochaine]

début ↑

Dernière mise à jour : 1997-10-24 Avis importants

[English] [Contactez-nous] [Aide] [Recherche] [Site du Canada]
[Accueil - SPSP] [Centres] [Publications] [Lignes directrices] [Indexe A-Z]
[Santé - Enfants] [Santé - adultes] [Santé - aînés] [Laboratoires] [Surveillance]

FIG. 4.1 – *Page web pour laquel OCELOT a généré un résumé.*

langues. Par exemple il peut d'une page Web française fournir un résumé en anglais.

Sur la Figure 4.1 nous pouvons voir une page Web résumée par OCELOT, et dont le résultat est le suivant:
«`health protection branch of the protection of health anti inflation guidelines health of animals in volume may table of contents of our children in central canada review of u.s beginning at page volume final vote day may`»

Le système est principalement basé sur des méthodes statistiques et il utilise des modèles probabilistes. Pour les traductions des méthodes en Natural Language Processing(NLP) sont utilisées. Nous n'entrerons pas dans les détails d'implémentation d'OCELOT sur la manière dont il résume une page Web (à ce titre voir [Berger and Mittal, 2000]). Mais examinons brièvement quelques étapes préliminaires appliquées sur les pages Web avant le processus de résumé proprement dit:

1. Normaliser le texte: enlever la ponctuation; remplacer toutes les lettres du texte en minuscule; remplacer les chiffres par le symbole NUM; enlever les 100 mots les plus communs;
2. Enlever tous les liens, les images et les méta-informations de la page HTML;
3. Enlever toutes les balises HTML.

Evaluation

Comme nous pouvons nous en rendre compte, OCELOT ne tient pas compte des informations encapsulées dans les balises HTML et des liens hypertextes. La raison est que, comme nous l'avons dit avant, la structure des pages Web n'est pas nécessairement homogène, et on ne sait pas affirmer que les informations contenues dans ces balises soient utilisées correctement. Toutes ces observations ont mené OCELOT à ne tenir compte que des mots.

4.6.2 Le Rôle des Liens Hypertextes

Le fait qu'OCELOT ne prenne pas en compte les liens hypertextes pour faire le résumé est selon [Barbosa, 2001] une faiblesse. En effet, un lien hypertexte a une valeur sémantique pouvant être exploitée.

Intuitivement, il est évident que si un auteur de pages Web y insére un lien vers un autre document c'est que ce dernier est intéressant. Mais est-il intéressant pour un résumé?

Afin de répondre à cette question il convient de caractériser les liens hypertextes comme suit:

1. *lien intérieur à la page vs lien extérieur à la page*: le premier correspond à un lien qui nous permet de naviguer sur la même page (on évite d'utiliser la barre de défilement). Tandis que le deuxième nous renvoie vers une nouvelle page;
2. *lien intérieur au site vs lien extérieur au site*: le premier renvoie vers une page du même site Web, tandis que l'autre renvoie vers une page extérieur au site;
3. *lien explicite vs lien implicite*: le premier est construit par l'auteur de la page Web, tandis que le second est généré automatiquement pendant la navigation.

Il va de soi que l'avantage ici est de prendre le contenu d'une page pointée par un lien pour créer le résumé. Il se peut que cette page ait un grand nombre de liens (plus de 10) et il s'agira probablement d'une page Web généré par un moteur de recherche. Dans ce cas les liens extérieurs peuvent être intéressants car, en principe, chaque lien pointe vers le même sujet. Inversément, pour ceux qui ont moins de 10 liens, on peut présumer que le résumé sera moins bon dû au manque de sources. Sauf si l'on pense que ces liens sont explicites. On peut donc voir l'avantage de la sémantique des liens hypertextes pour la constitution d'un résumé.

Améliorer la qualité des résumés sur le Web

Toujours selon [Barbosa, 2001], pour améliorer la qualité des résumés sur le web, il faut considérer les points clés suivants:

1. Les mécanismes doivent comprendre un peu mieux le contexte afin d'eviter le plus possible les ambiguïtés du langage naturel. Par exemple OCELOT n'est pas capable de faire la différence entre «le soleil éclaire la terre» et «la terre éclaire le soleil»;
2. Dans certaines application on peut produitre un résumé du type *user-specific*: prendre en compte la requête de l'utilisateur pour la constitution du résumé;
3. Prendre en compte la diversité des langues est indispensable;

4. Contrairement à OCELOT il faut tenir compte des balises HTML et s'en servir comme méthode du «lexical cue» (voir Section 4.4). Avec les balises $< \backslash TITLE > \cdots < \backslash TITLE >$ on peut par exemple identifier les phrases importantes de la page Web.
5. Et finalement, la sémantique des liens hypertextes.

4.7 Application aux PDAs

A présent, ayant vu les différentes notions associées aux résumés, ainsi que quelques techniques pour créer des *extracts* et la manière dont on peut les appliquer aux pages Web, nous allons examiner comment ces principes peuvent être augmentés pour s'adapter aux PDAs. Nous allons voir cela à travers l'exemple du Power Browser (voir [Buyukkokten et al., 2001]).

4.7.1 Power Browser

Principe

Le principe du Power Browser est que si l'utilisateur en phase de recherche d'informations désire visualier une page Web alors cette recherche peut se faire de manière progressive. La particularité de ce système est qu'il ne résume pas l'entiéreté de la page Web mais plusieurs portions.

Le Power Brower fournit à l'utilisateur la possibilité de ne voir qu'une partie de la page Web à travers son système d'«Accordion Summarization». Pour ce faire, la page est divisée en «Semantic Textual Units» (STU) représentant des morceaux logiques de la page Web (voir Figure 4.2). L'idée est de ne montrer que des portions des STUs par l'intermédiaire de résumés.

Afin de déterminer les STUs, la méthode du «Lexical Cue» (voir Section 4.4) est utilisée. Nous rappelons que cette méthode détermine les phrases importantes en fonction de certains *cue words* identifiés heuristiquement. De manière analogue, un certain nombre d'heuristiques relatives aux pages HTML sont utilisées pour déterminer les *cues* e.g. la présence de balises telles que $< \backslash TITLE > \cdots < TITLE >$.

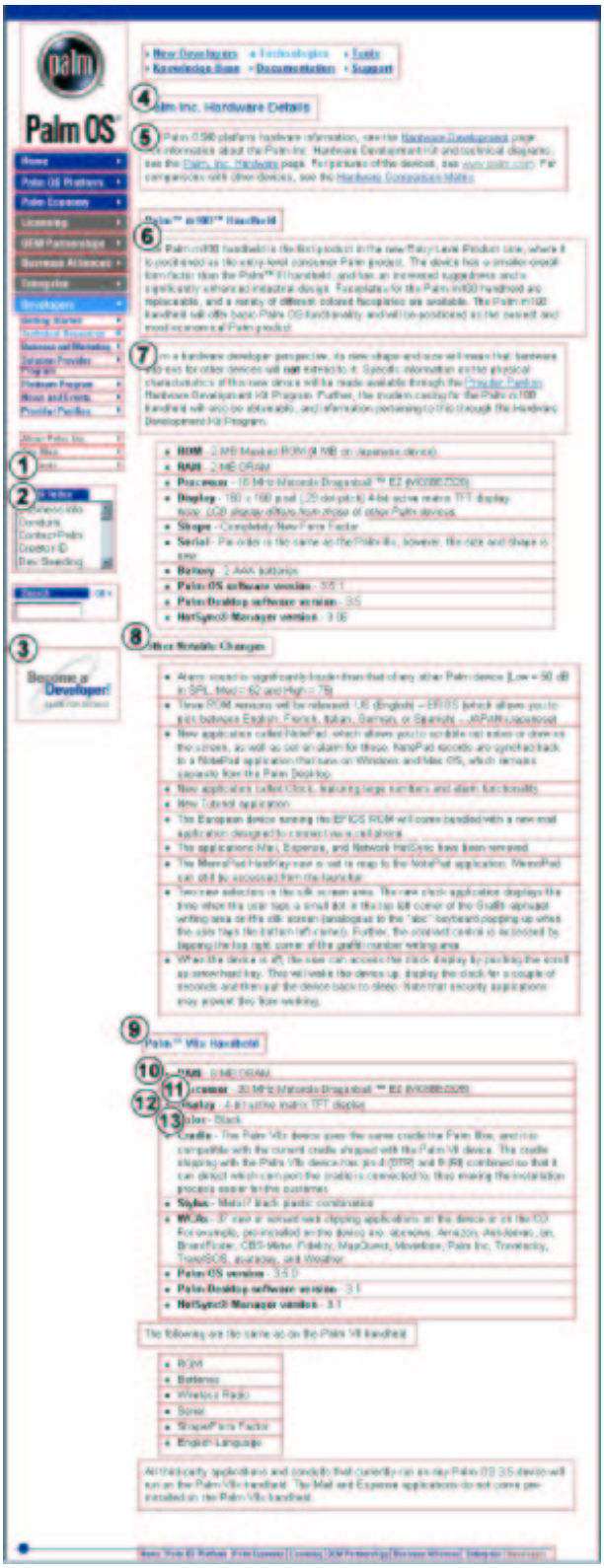

FIG. 4.2 – *Exemple de STUs pour la page: chaque STU est numéroté de 1 à 13. Web de PalmOS*

Téchniques utilisées

Afin de résumer chaque STU quatre méthodes sont utilisées (voir Figure 4.3). La première (i.e. *incremental*), est en fait la méthode de «Location» (voir Section 4.4): pour chaque STU on montre la première phrase. Ceci permet à l'utilisateur de savoir si ce STU est susceptible de l'intéresser. Si l'utilisateur sélectionne le cercle qui se trouve devant la phrase sur l'écran, alors les deux premières lignes du STU seront visibles. Il faut deux interactions pour découvrir l'entiéreté du STU.

La deuxième (i.e. *keyword*) est la méthode des *mots clés* TF/IDF à ceci près que la collection d'où vient le document est le World Wide Web et des approximations sont donc utilisées afin d'approcher la valeur de N et de n, respectivement le nombre de documents dans la collection et le nombre de documents dans lequel le mot apparaît au moins une fois (voir équation [4.1] à la page 49). Dans ce cas, il faut également deux étapes pour voir l'entiéreté du STU.

La troisième méthode (i.e. *summary*) est basée sur la méthode de fréquence des mots de Luhn à ceci près que l'importance des mots est évaluée grâce à la méthode TF/IDF (voir Section 4.4). Ici, une étape est nécessaire pour voir l'entiéreté du STU.

Et finalement, la dernière (i.e *keyword/summary*) mixe les deux dernières méthodes.

Nous pouvons observer sur la Figure 4.4 un exemple concret de mise en oeuvre des différentes techniques. Pour des raisons de présentation la méthode «*All*» est sur deux colones.

Chaque entrée d'un STU est associé à un cercle de cinq pixels (noir, noir et blanc, et blanc). il sert à contrôler l'affichage du STU et permet à l'utilisateur de savoir combien d'étapes il lui reste à effectuer avant de voir l'entiéreté du STU. Notons qu'un certains nombre de mécanismes ont été mis au point afin que le processus d'ouverture et de fermeture des STUs soit ergonomique.

Implémentation Tout le processus est créé dynamiquement sur un proxy qui effectue tous les traitements des pages Web et qui les fournit aux PDAs. Le proxy filtre également le contenu des pages Web afin qu'elles ne contiennent pas d'élements non nécessaires (e.g. Multimédia). L'utilisateur peut donc visualiser n'importe quelle page du World Wide Web de cette manière.

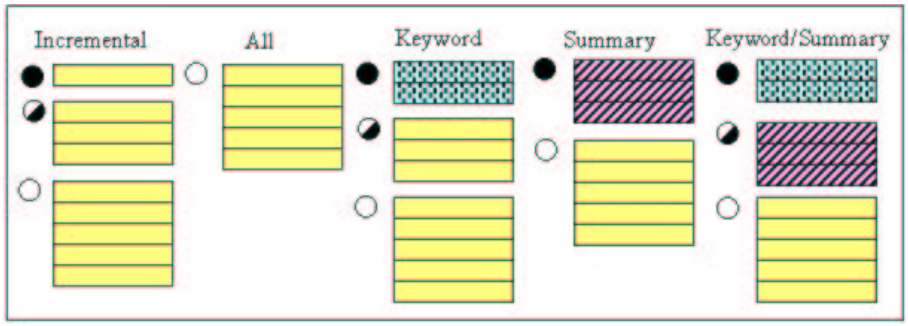

FIG. 4.3 – *Les méthodes pour découvrir l'ensemble des STUs.*

Evaluation Comme nous pouvons le voir les techniques proposées sont basées sur des méthodes statistiques de l'Information Retrieval. La manière dont le Power Browser traite les liens hypertextes dépend de la technique utilisée. Par exemple la méthode *Keyword* ne montre pas du tout les liens hypertextes. Tandis que la méthode *Incremental* les montre et elle nous permet de les sélectionner s'ils commencent le début d'un STU. Chaque méthode a des caractéristiques différentes pour traiter les liens hypertextes.

Le système qui montre la meilleure performance est celui qui utilise la méthode mixte *keyword/summary*.

4.8 Conclusions

Nous avons vu dans ce chapitre les différents éléments qui sont à la base des systèmes IR, ainsi que la manière dont on pouvait évaluer leur qualité. Nous avons expliqué un phénomène propre aux systèmes IR en introduisant les notions de *precision* et de *recall*, et examiné en quoi les techniques de génération automatique de résumé pouvaient en réduire les effets négatifs.

Comme nous l'avons déjà dit, les bibliothèques digitales offrent par nature un système IR. De ce système dépend en partie la qualité de l'information que l'on trouve. Par conséquent, d'un bon système IR dépendra également la qualité de la solution offerte par GreenstoneToAvantgo. Car en offrant un accès exclusif aux bibliothèques digitales (construites par le software Greenstone), elle favorise la découverte d'informations de qualité.

Nous avons également mis en avant les difficultés pour générer automatiquement un résumé de pages Web. Bien que GreenstoneToAvantGo ne suit

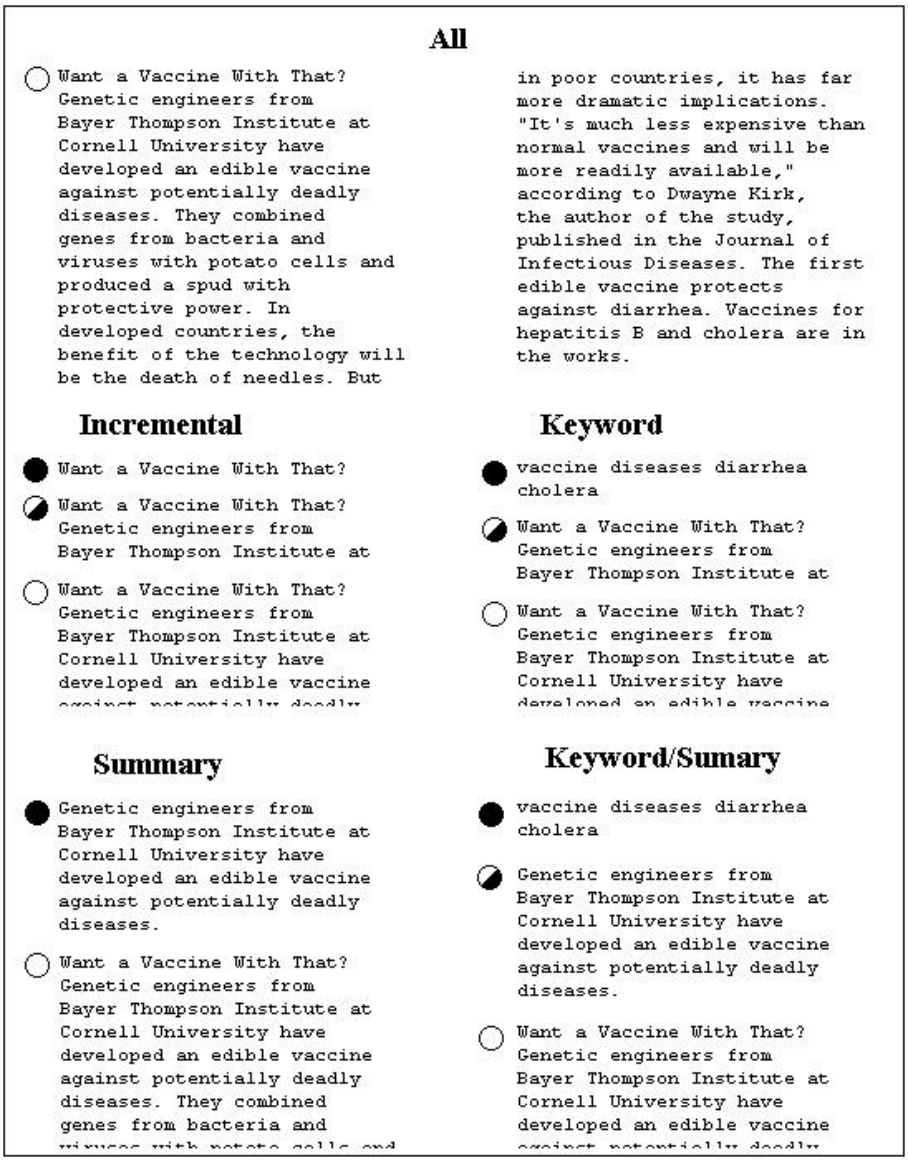

FIG. 4.4 – *Exemple des méthodes pour découvrir l'ensemble des STUs.*

pas l'approche du Power Browser, il serait possible, à l'instar de ce dernier, de partager les pages Web en STUs et d'appliquer une variante de la méthode *incremental* en montrant 2 ou 3 lignes.

De plus il existe en générale dans les bibliothèqes digitales un certain nombre de documents qui sont fournis avec un résumé fait par l'homme. Il suffirait donc à GreenstoneToAvantgo de les extraire afin de les fournir aux utilisateurs.

Le challenge de ces deux solutions est l'implémentation d'un parseur suffisamment puissant pour déterminer les STUS et/ou pour extraire les résumés produits par l'homme.

Deuxième partie

Solution proposée par GreenstoneToAvantgo

Chapitre 5

Greenstone Digital Library

Le but de ce chapitre est de présenter le software Greenstone[1], car le système GreenstoneToAvantgo (que nous présenterons au Chapitre 6 se base principalement su lui[2].

Nous verrons à la Section 5.2 en quoi les bibliothèques digitales peuvent aider les pays en voie de développement. Rappelons que l'un des objectifs de notre application est d'offrir l'accès aux bibliothèques digitales de Greenstone aux pays en voie de développement.

Nous examinerons à la Section 5.3 l'organisation de la bibliothèque digitale de Greenstone. A l'image des bibliothèques traditionnelles il existe une organisation spécifique.

A la Section 5.4 nous identifierons les fonctions de recherches qu'offre la bibliothèque. Principalement, la recherche par mots clés et la navigation.

Nous verrons à la Section 5.5 quelques d'exigences auxquelles répondent les bibliothèques digitales de Greenstone. Malheureusement, nous n'approfondirons pas ce thème. Bien qu'intéressant il dépasse le cadre de ce mémoire et demanderait un mémoire à lui seul.

Ensuite, nous aborderons le processus de construction de bibliothèques digitales à la Section 5.6. Nous verrons comment le système s'y prend pour créer

[1] C'est dans le cadre du *New Zealand Digital Library Project* (NZDLP, `http://www.greenstone.org` en collaboration avec l'UNESCO (`http://www.unesco.org`) et le Human Info (`http://humaninfo.org`) qu'il a été concu. Le but de Greenstone est de permettre à n'importe quelle personne ou institution de construire sa propre bibliothèque digitale.

[2] En effet, celui-ci a pour objet de transformer les pages Web des bibliothèques digitales de Greenstone et de les mettre dans un format approprié pour les visualiser sur les PDAs

des collections ainsi que toutes les structures de données qui permettent de supporter les fonctions de la bibliothèque digitale.

Et pour finir, nous jetterons un regard à l'implémentation du système Greenstone à la Section 5.7. Cette section nous permettra de comprendre quelles sont les aspects techniques qui entrent en compte dans l'implémentation de GreenstoneToAvantGo.

5.1 Introduction

Jusqu'à nos jours l'humanité a accumulé une quantité d'informations considérable dans les Arts, les Sciences,etc., mais également au cinéma, en musique et en télévisuel. Ce phénomène pose le problème de leurs accès.

Les récentes avancées en matière de stockage de données et de digitalisation ont fait de l'archivage digitale un moyen possible et peu coûteux. De plus selon [Lyman and Varian, 2000] le monde produit entre un et deux exabytes d'information chaque année. Ce qui est à peu près 250 megabytes par habitant dans le monde. La plupart de ces informations sont sous forme d'images, d'audio et de documents numériques. La part des documents imprimés ne représentent que 0.003% du total.

Afin qu'un maximum de personnes puisse avoir accès à ces informations, il est nécessaire de réorganiser l'information fournie sur l'Internet. Sans quoi ce riche potentiel ne profitera qu'à un nombre limité de personnes.

C'est à cette problématique que répondent les bibliothèques digitales: organiser l'information, la rendre accessible depuis le World Wide Web, la maintenir à travers les années, gérér des données non conventionnelles, des collections multiculturelles et multilinguistiques.

5.2 Les Bibliothèques digitales et les pays en voie de développement

[Witten et al., 2001] identifie cinq champs à travers lesquels les bibliothèques digitales peuvent améliorer le niveau de vie des pays en voie de développement.

Dissémination d'informations humanitaires

Les réseaux de distribution et de publication n'ont pas fonctionné correctement pour fournir de l'information médicale à tous de manière équitable. Par exemple, la bibliothèque médicale du *Nairobi University Medical School Library* considéré longtemps comme la plus riche d'Afrique de l'Est recevait 20 journaux en l'an 2000 (à comparer avec les 300 qu'elle recevait il y a 10 ans). La même année, n'importe quelle bibliothèque médicale américaine était abonnée à 5000 journaux. Un autre exemple est l'Université du Congo Brazzavile où l'on ne peut trouver que 40 livres de médecine et une douzaine de journaux, tous publiés avant 1993.

Les bibliothèques digitales permettent de réduire les coûts de distribution de l'information. New Zealand Digital Library Project met à disposition de l'information humanitaire en offrant près de 3590 publications. La version papier de la collection Human Development Library, par exemple, contient 1230 publications, pèse 340kg et coûte 20 000$.

Reconstruction après un désastre naturel

Après un désastre naturel tel un tremblement de terre, un ouragan, ou autre, le besoin d'informations est immédiat. Une information variée doit être disponible rapidement aux différents acteurs qui entrent en jeu dans la phase de reconstruction.

Grâce aux technologies des bibiliothèques digitales on peut rapidement créer et organiser des collections d'informations, qui combinent des informations de médecine générale et sanitaire, tout en répondant aux besoin d'un désastre spécifiques. En effet, les collections peuvent tenir compte de la nature de la région touchée et des ressources logistiques disponibles. De plus elles possédent des outils de recherche d'informations très efficaces.

Préserver et propager sa culture

Les deux précédents points suggéraient le transfert de la connaissance des pays développés vers les pays en voie de développement. Chaque pays possède une connaissance très riche. L'information culturelle peut prendre plusieurs formes: histoires orales, images, musique, partitions de musique, paroles, dances et cérémonies sous forme de vidéos, d'audio, etc. Aider les pays en

voie de développement à créer leurs propres collections d'informations est une stratégie plus efficace pour supporter une croissance durable.

Les bibliothèques digitales supportent plusieurs langues et offrent des possibilités multimédia.

Produire des collections d'informations

Les personnes des pays en voie de développement ont une connaissance médicale basée sur les plantes locales. Ils ont également une connaissance de longue date sur la façon de cultiver, et de protéger des espèces animales et végétales.

Leur donner la possibilité de rassembler leurs connaissances dans de grandes collections est primordiale. De plus, cela leur permettrait de partager cette connaissance avec les autres pays, notamment les pays industrialisés.

Nouvelles opportunités d'entrer sur le marché international

Bon nombre de documents se trouvent encore sous forme papier. Quelques pays dont l'Inde, la Roumanie et les Philippines ont entrepris la conversion de documents papiers sous forme digitale.

Le développement des bibliothèques digitales demande un certain effort manuel.Des tâches telles l'organisation des collections de documents, par exemple, sont facilement faisables par les pays en voie de développement. Ils peuvent de la sorte offrir de nouveaux types de service au monde entier.

5.3 Organisation de Greenstone Digital Library

Sur la «home page» de la bibliothèque de Nouvelle-Zélande (voir Figure 5.1), nous pouvons voir quelques *collections* identifiées par une image, et un titre, e.g. la première collection en haut à gauche se nomme «Humanity Developpement Library». Chaque collection est accompagnée d'une description de ses buts, ainsi que de la manière dont elle est organisée (voir Figure 5.2) ([Witten and Boddy, 2001b]).

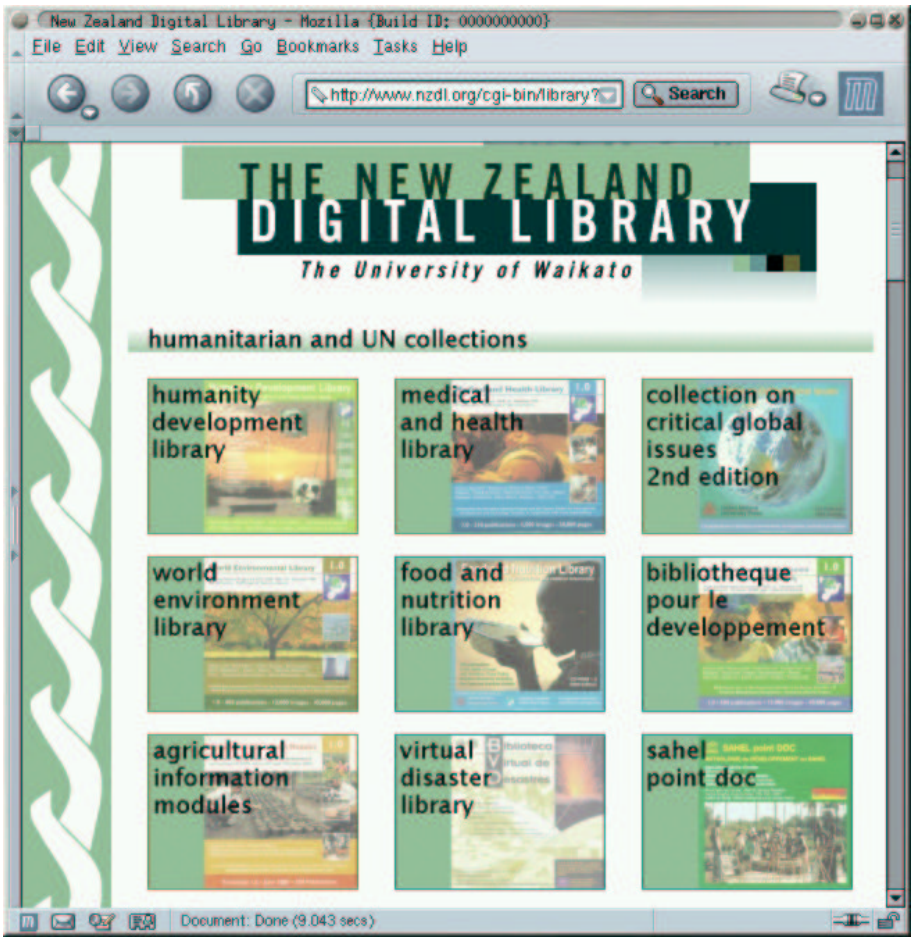

FIG. 5.1 – *La home page de la bibliothèque digitale de Nouvelle-Zélande.*

FIG. 5.2 – *La page d'acceuil d'une des collections de la bibliothèque digitale de Nouvelle-Zélande.*

Une collection contient plusieurs (typiquement quelques milliers, voire quelques millions) de *documents*. Il existe une interface uniforme à tous les documents d'une collection. Chacune de ces collections peut être organisée différement — bien qu'il existe une ressemblance manifeste dans la manière dont elles sont présentées.

Un document peut être un livre. Dans ce cas, l'image de la couverture du livre, ainsi que sa table des matières seront visibles. Dans certains cas un résumé sera disponible (voir Figure 5.3). Chaque livre est lui même organisé de manière hiérarchique en *sections*, *sous-sections*, et *paragraphes*.

5.3.1 L'interface utilisateur

Un soin particulier a été pris pour offrir à l'utilisateur une interface de qualité. Toutes les icônes des captures d'écran peuvent être sélectionnées en cliquant dessus.

Les trois icônes sur le haut de la page (*HOME* ... *HELP* ... *PREFERENCES*) permettent:

- de revenir sur la «home page»;
- d'avoir un texte d'aide;

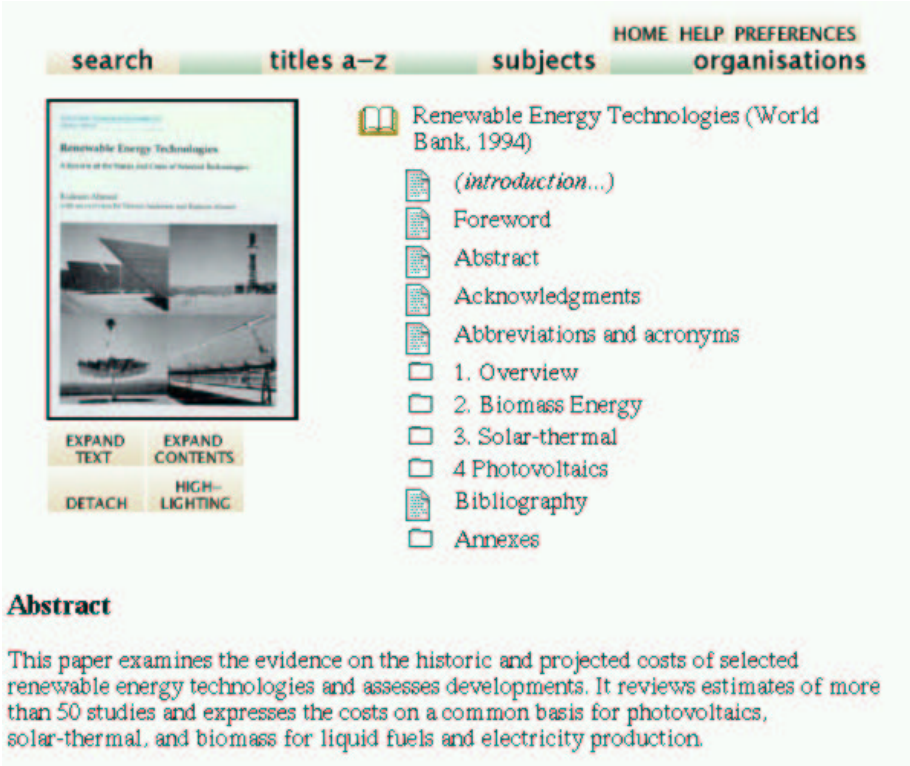

FIG. 5.3 – *Exemple d'un livre: la table des matières et un résumé.*

– de changer les préférences de l'interface (e.g. changer la langue) et de définir les préférences de recherche.

La barre de navigation *search ... title a-z ... subjects ... organisations* donne accès aux outils de recherches par mots clés, et permet de visualiser les documents d'une collection sous une autre classification.

Au dessous de l'image du livre de la Figure 5.3 à la page 71, il y a quatre icônes permettant de contrôler l'affichage du livre. Il est possible de montrer la table des matières de manière détaillée (toutes les sections et toutes les sous-sections), ainsi que de montrer le document dans son entiéreté dans la fenêtre du navigateur (ce qui est pratique pour imprimer). Il est également possible de voir une section, ou une sous-section, en cliquant sur l'icône la représentant (i.e. un dossier).

L'utilisateur peut toujours savoir à quel niveau il se trouve dans le livre, grâce aux icônes qui ont la forme d'un livre ou d'un dossier ouvert ou fermé. De plus le titre de la section lue est mis en gras dans la table des matières. A la fin de la page, deux icônes représentant une flèche permettent à l'utilisateur d'avancer ou de reculer dans le livre.

5.4 Trouver de l'information

Les collections d'informations construites par Greenstone combinent la recherche par mots clés et la navigation basée sur différentes méta-informations (e.g. auteurs, titres d'oeuvre, dates). L'utilisateur a donc plusieurs possibilités pour la recherche de documents; tout dépendant du design de la collection et des méta-informations disponibles.

5.4.1 Recherche par mots clés

Typiquement, il est possible de faire la recherche d'un mot particulier qui apparaît dans un texte, à l'intérieur d'une section, ou simplement à l'intérieur d'un titre ou d'un sous-titre du document. En effet, dans certaines collections il existe un index des mots pour chacune de ces divisions.

Nous pouvons voir sur la Figure 5.4 un exemple de recherche du mot «chance» dans les chapitres des documents de la collection World Environnement Library (WEL). Les titres des 20 premiers documents qui concordent

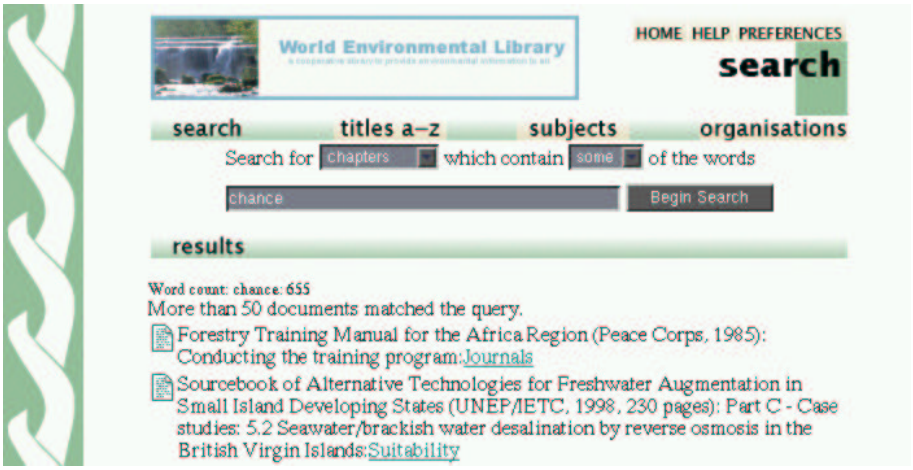

FIG. 5.4 – *Exemple des résultats de la recherche pour le mot chance.*

avec la requête sont montrés. Il est possible de visualiser les 20 résultats suivants en cliquant sur la flèche présente au bas de la page. Pour des raisons d'efficacité un maximum de 100 documents retournés est imposé. En cliquant sur le titre il sera possible de voir le document, une section ou une sous-section du document.

L'interface utilisateur par défaut pour la recherche d'informations est simple. Elle dispose de deux types de requêtes:

1. les requêtes pour lesquelles tous les mots spécifiés par la recherche se trouvent repris dans chaque document des résultats.
2. les requêtes pour lesquelles les mots spécifiés par la recherche ne sont pas repris dans tous les documents des résultats. Dans ce cas les titres de documents sont listés dans un ordre qui est défini par la proximité des termes présents. De plus, pour déterminer le degré de concordance avec les mots de la recherche, on regarde:
   (a) les documents contenant le plus grand nombre des termes de la requête;
   (b) les termes les plus rares. Ils sont les plus importants;
   (c) les petits documents qui sont préférés aux grands documents.

Il est possible d'utiliser autant de termes de recherche que désirés. Si l'on spécifie un seul terme, il est possible d'ordonner les résultats en fonction des documents contenant le plus fréquemment le mot de la recherche; pour cela

il faut choisir la méthode *some* à la place de *all* dans les options de recherche disponibles sur l'interface.

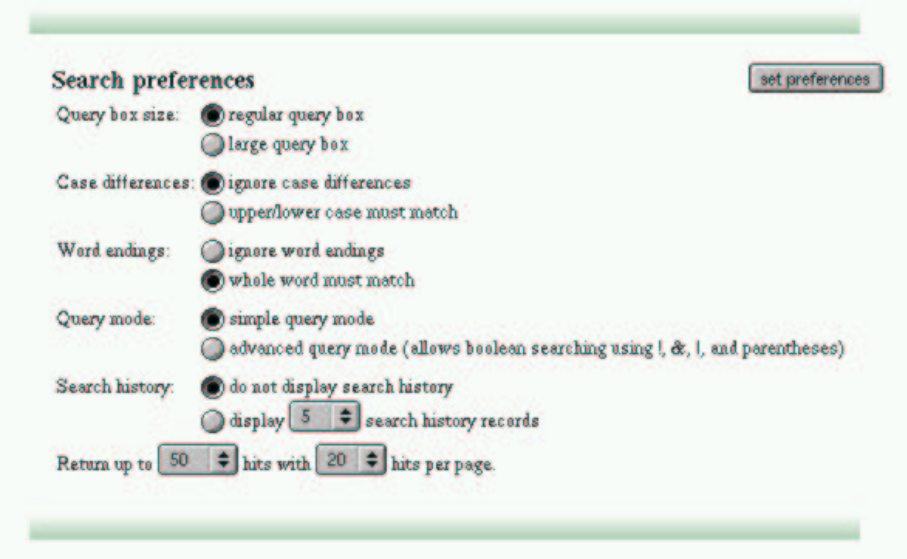

FIG. 5.5 – *Exemple de l'interface fournie pour les utilisateurs expérimentés dans la recherche de documents.*

Afin de répondre aux besoins de recherches plus exigentes, une interface pour les utilisateurs expérimentés est disponible (voir Figure 5.5). On peut autoriser de:

- considérer les majuscules et les minuscules;
- ne considérer les termes qu'en fonction de leur radical;
- définir des opérateurs booléens;
- finalement, obtenir un historique des recherches précédentes (cela facilite la répétition de requêtes sensiblement différentes).

5.4.2 Naviguer au travers des collections

Pour la navigation, chaque collection offre plusieurs types de classification différentes. Elles sont basées sur des méta-informations telles les auteurs, les titres, les dates, les mots clés, etc. Il est donc possible d'alterner la vision que l'on a de la collection en sélectionnant un autre type de classification. Par exemple, sur la Figure 5.2 de la page 70, on peut voir que cette collection posséde trois manières de classifier les documents. Il est donc possible

d'accéder aux publications classées par *sujet* en cliquant sur le bouton *sujets*; accéder aux publications classées par *titre* en cliquant sur le bouton *titles a-z*, qui fournit une liste de livres classés par ordre alphabétique; et finalement, accèder aux publications classées par *organisation* en cliquant sur le bouton *organisations*, qui fourni une liste d'organisation.

Sur la Figure 5.6 on peut voir comment naviguer dans la collection lorsque les documents sont classés par sujet: en cliquant sur l'image du *bookshelf* (icône représentant un petit ensemble de livres) l'utilisateur peut découvrir une liste de livres représentés par l'image du *book* (icône représentant un livre) (voir Figure 5.7). Il peut visualiser un livre en cliquant sur l'icône le représentant (voir Figure 5.3 à la page 71 pour un exemple de livre).

FIG. 5.6 – *Exemple des sujets disponibles dans une collection.*

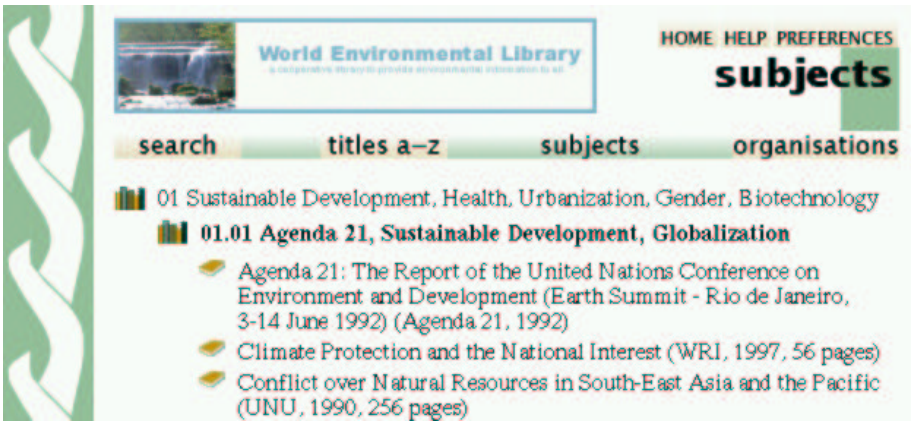

FIG. 5.7 – *Exemple des livres disponibles dans une sous-collection classée par sujet.*

5.5 *Digital Library Requirements*

Les exigences sur les bibliothèques digitales sont nombreuses et Greenstone Digitale Library y répond. Nous allons vous en présenter quelques unes par la suite. Nous n'appronfondirons pas ces questions, car bien qu'intéressantes elles dépassent le cadre de ce mémoire.

Maintenance

Greenstone facilite la *maintenance* du système en créant ses structures de données automatiquement à partir des documents eux-mêmes (e.g. aucun lien hypertexte n'est ajouté à la main). Cela signifie que n'importe quel document peut être ajouté automatiquent dans la structure existante. Pour certaines collections cette mise à jour est faite régulièrement par la recherche de nouveaux documents sans intervention de l'homme.

Support multiformat & multimédia

Afin de fournir de nombreux types différents de documents (e.g. les extentions pdf ou ps), le software est organisé de telle manière qu'il est possible d'ajouter des «plugins» pour chaque format de documents. Ces documents sont traités par les plugins pour être sauvés dans un nouveau format afin qu'ils soient visualisables sur le World Wide Web. Une collection peut ainsi contenir de nombreux types de documents: il suffit de spécifier les plugins nécessaires.

Les collections peuvent contenir du texte, des images, et même des sons et des clips vidéo. La plupart des documents non-textuels sont décrits afin d'être disponible pendant la phase de recherche par mots clés.

Support Multilinguistique et multiculturel

Le code international Unicode est utilisé: les documents — et les interfaces utilisateurs — peuvent être écrits dans n'importe quelle langue. Actuellement, des collections ont été produites en Anglais, en Francais, en Néerlandais, en Portugais, en Italien, en Espanol, en Allemand, en Maori, en Chinois, en Arabe, en Russe, en Indonésien et en Hebreu. Le site Web de Nouvelle-Zélande offre de nombreux exemples. Une interface en mode texte est également disponible. Ceci permet de réduire la taille des pages Web à télécharger.

Sécurité

Le système inclut une fonction «administrative» grâce à laquelle les administrateurs de la bibliothèque peuvent vérifier la composition des collections et protéger certains documents. Il est possible de définir des groupes d'utilisateurs par le biais de logins et de mots de passe, etc. Il existe également un fichier journal sur lequel toutes les activités des utilisateurs relatives à leurs requêtes sont enregistrées (cette fonction peut être désactivée).

Transparence

Le Software Greenstone fonctionne sous Unix et Windows NT, et avec des serveurs Web standards. Une structure de processus flexibles permet à différentes collections d'être servies par différents ordinateurs, tout en laissant cette mise en oeuvre transparente pour l'utilisateur: la collection est présentée à l'utilisateur toujours de la même manière (e.g. même page Web, même bibliothèque digitale). Des collections existantes peuvent être mises à jour, et de nouvelles collections peuvent être mises en ligne en même temps sans arrêter le serveur Web. Le processus responsable de l'interface utilisateur va s'en rendre compte, et les nouvelles collections apparaîtront et seront ajoutées à la liste des collections existantes.

5.6 Construction de la bibliothèque

Toute la structure de navigation et les index sont créés pendant la phase d'*importing*. C'est à ce moment que de nouveaux documents sont ajoutés aux collections ([Witten and Boddy, 2001a]).

5.6.1 Le processus d'*importing*

La première des responsabilités du processus d'*importing* est de convertir les documents d'origine dans le format GML (Greenstone Markup Language) utilisé à l'intérieur de Greenstone. Le processus de conversion est rempli par les plugins associés au format des documents.

Le language GML (basé sur XML[3]) définit (voir Figure 5.8):

- la structure des documents (section, sous-sections, etc);
- les méta-informations associées aux documents;
  - l'identifiant du document (OID [Object Identifier]) dans le système (dans le fichier de la Figure 5.8, il se nomme Identifier);
  - et d'autres méta-informations.

```
<gsdlsection>
  <metadata>
    <gsdlsourcefilename>/home/gsdl/collect/geonet/tmp/fFaggot'sfretfulfiascos,html</gsdlsourcefilename>
    <gsdldoctype>indexed_doc</gsdldoctype>
    <Language>en</Language>
    <Encoding>iso_8859_1</Encoding>
    <Subject>3</Subject>
    <Creator>Professor 3</Creator>
    <Title>Faggot's Fretful Fiascos</Title>
    <Date>19951004</Date>
    <srclink>&lt;a href=_httpcollection_/index/assoc/[archivedirl/doc,rtf&gt;</srclink>
    <srcicon>_iconrtf_</srcicon>
    <&#47;srclink>&lt;/a&gt;</&#47;srclink>
    <Identifier>HASH011e97d99f4d11e4dc44ceac</Identifier>
    <gsdlassocfile>doc,rtf:application/rtf:</gsdlassocfile>
    <assocfilepath>HASH011e,dir</assocfilepath>
  </metadata>

La source du texte

</gsdlsection>
```

FIG. 5.8 – *Exemple de méta-informations d'un fichier GML. Chaque section peut également contenir des méta-informations.*

Chaque document est identifié de manière unique grâce à un OID, qui est en fait le code de *hashing* du document. Chaque section dans un document est également identifiable de manière unique par l'OID du document auquel on ajoute un suffixe (à l'aide d'un point et d'un chiffre).

Les méta-informations sont des informations descriptives telles les auteurs, les titres, des mots clés, etc. Le *Dublin Core metadata standard* est utilisé

[3]XML (eXtensible Markup Language) est en quelque sorte un langage HTML amélioré permettant de définir de nouvelles balises. Il s'agit effectivement d'un langage permettant de mettre en forme des documents grâce à des balises (markup).

Contrairement à HTML, qui est à considérer comme un langage défini et figé (avec un nombre de balises limité), XML peut être considéré comme un métalangage permettant de définir d'autres langages, c'est-à-dire définir de nouvelles balises permettant de décrire la présentation d'un texte. La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il va permettre de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir.

En réalité les balises XML décrivent le contenu plutôt que la présentation (contrairement à HTML). Ainsi, XML permet de séparer le contenu de la présentation. Ce qui permet par exemple d'afficher un même document sur des applications ou des périphériques différents sans pour autant nécessiter de créer autant de versions du document que l'on nécessite de représentations [Pillou, 2002].

Name	Metadata subtag	Definition
*Title	*Title*	A name given to the resource
*Creator	*Creator*	An entity primarily responsible for making the content of the resource
*Subject and keywords	*Subject*	The topic of the content of the resource
*Description	*Description*	An account of the content of the resource
*Publisher	*Publisher*	An entity responsible for making the resource available
Contributor	*Contributor*	An entity responsible for making contributions to the content of the resource
*Date	*Date*	The date that the resource was published or some other important date associated with the resource.
Resource type	*Type*	The nature or genre of the content of the resource
Format	*Format*	The physical or digital manifestation of the resource
*Resource identifier	*Identifier*	An unambiguous reference to the resource within a given context: this is the object identifier or OID
*Source	*Source*	A reference to a resource from which the present resource is derived
*Language	*Language*	A language of the intellectual content of the resource
Relation	*Relation*	A reference to a related resource
*Coverage	*Coverage*	The extent or scope of the content of the resource
Rights management	*Rights*	Information about rights held in and over the resource

FIG. 5.9 – *Dublin Core metadata standard.*

pour définir les types des méta-informations: la Figure 5.9 nous en donne une description. Evidemment, il est possible d'utililiser de nouveaux types de méta-informations.

5.6.2 Le processus de construction

Durant cette phase les documents sont compressés, stockés dans une collection de documents, et l'index portant sur l'ensemble du texte est construit. En fonction du fichier de configuration, l'index peut être raffiné pour prendre en compte différentes sections, sous-sections ou paragraphes d'un document. La structure nécessaire à la navigation est créée grâce à un composant nommé *classifiers*, qui utilise les méta-informations contenues dans les fichiers GML.

5.6.3 Le fichier de configuration

Toutes les informations nécessaires à la construction d'une collection (e.g. plugins, classifier) sont enregistrées dans un fichier de configuration. Il définit aussi d'autres options.

5.6.4 Génération des pages Web

Les pages Web sont générées «à la volée» pendant la navigation. Quelques aspects des pages sont contrôlés en utilisant des *formats strings* (qui sont également définis dans le fichier de configuration). Deux types différents de pages Web sont contrôlés par les *formats strings*. La première définit les pages qui montrent un document ou une partie d'un document. La seconde définit les pages Web qui montrent des listes de documents.

L'ensemble de l'interface de Greenstone est contrôlé par des *macro files*. Ces *macros files* sont écrits dans un langage spécialement conçu pour Greenstone, et sont utilisés au moment de la génération des pages Web. Greenstone traduit le langage des *macros files* en HTML afin que l'internaute puisse voir la page Web.

Il y a de nombreuses raisons pour lesquelles les pages sont générées «à la volée». Il est possible de définir l'apparence complète de l'interface utilisateur, y compris la langue (ce qui est très important pour supporter le multilinguisme), les couleurs, les images, etc., grâce aux *macros files*. Si un utilisateur chinois se connecte à la bibliothèque digitale de Nouvelle-Zélande il peut souhaiter consulter l'interface en chinois. Grâce aux *macros files* definis pour l'interface chinoise le système en générera une. Il est donc possible de servir plusieurs personnes de culture différente au même moment sans rien toucher au système. En définissant de nouveaux *macros files* on ajoute le support d'une langue supplémentaire.

5.7 *The Greenstone Runtime System*

5.7.1 Architecture

Dans sa version simple le système Greenstone fonctionne comme sur la Figure 5.10. Il existe deux composants importants. Le *receptionist* reçoit les inputs (e.g clic de souris sur l'interface) des utilisateurs qu'il fait suivre par le biais d'un protocole commun au *collection server*. Le *collection server* localise les informations demandées dans le système et les renvoie au *receptionist* qui les fournit à l'utilisateur. Le *collection server* fonctionne comme un mécanisme qui manipule le contenu des collections, tandis que le *receptionist* est responsable de l'interface utilisateur.

Le composant *receptionist* est aidé des *macros files* et des *format string*

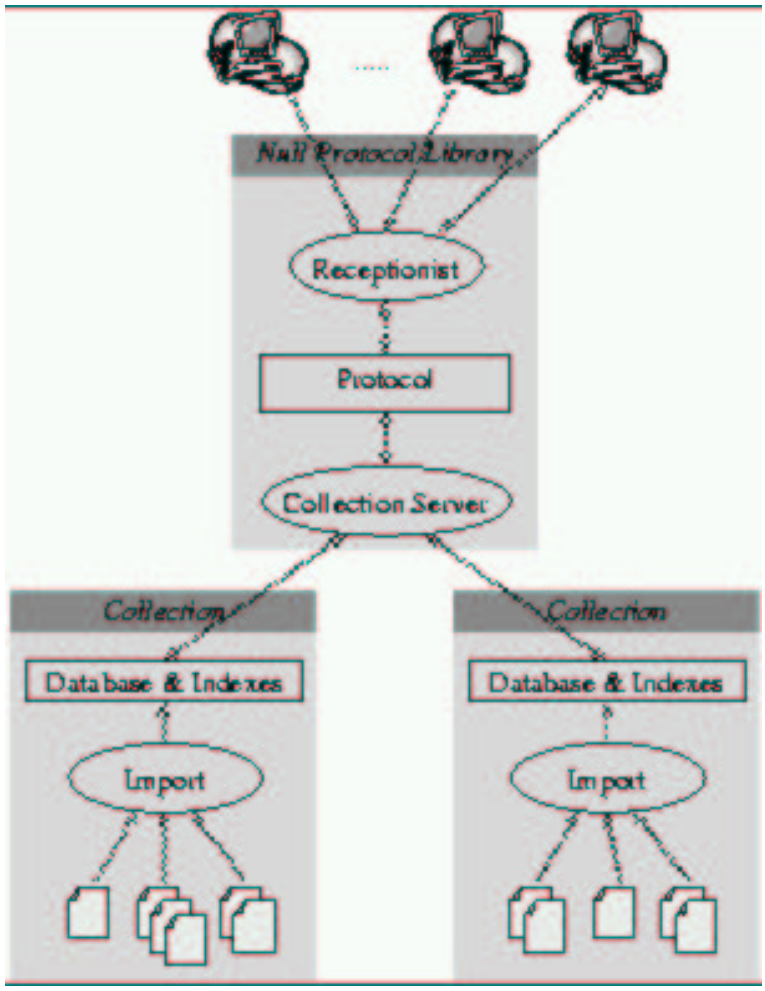

FIG. 5.10 – *Greenstone runtime system utilisant le «null protocol».*

pour générer l'interface. Le composant *collection server* fonctionne à l'aide de deux programmes précompilés, MG (i.e.*a full text retrieval system*) pour la recherche d'informations, et GDBM (i.e. *a database management system*) qui maintient la base de données des informations relatives aux collections.

Afin d'encourager l'extension et la flexibilité du système, Greenstone utilise le mécanisme de l'héritage (à travers le language dans lequel il est implémenté [i.e. C++]). Cela implique que les deux programmes MG et GDBM peuvent facilement être remplacés s'il le faut. De plus l'architecture du software est suffisamment riche pour pouvoir supporter complètement les possibilités multimédia telles le contrôle de l'interface à travers le «speech input».

Dans la pratique, les deux processus sont combinés pour former un exécutable nommé *library*. On parle dans ce cas de *null protocol*. Chaque fois qu'une page Web est demandée, le programme *library* est démarré (par le mécanisme CGI[4]), répond à la demande et puis s'arrête.

5.7.2 Le programme *library*

Afin de laisser la discussion à un niveau élémentaire les exemples ci-dessous ont été simplifiés.

Récupération de documents

Lorsqu'un document doit être récupéré dans la collection, la page correspondante est générée en faisant appel au program CGI *library* (voir Figure 5.11 [regardez la barre d'URL]). Les arguments que l'on passe sont spécifiés notamment par `a=d` qui indique que le système doit construire une nouvelle page. La variable `d` prend comme argument l'OID du document qui va être généré, par exemple `d=HASH01fd8570250e.1` (comme nous pouvons le voir, ce document est suffixé par `.1` ce qui correspond à la première section du document identifié par `d=HASH01fd8570250e`).

[4]Un CGI (Common Gateway Interface) est un programme exécuté sur le serveur de pages Web. Il permet de générer des pages dynamiquement en fonction de certains paramétres. Ces paramètres sont envoyés à l'application sur le serveur par le biais de l'Internet.

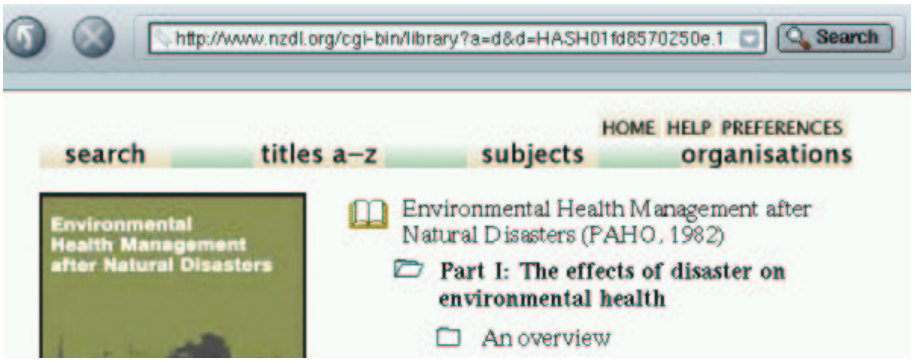

FIG. 5.11 – *Le programme CGI «library» pour retirer un document.*

Naviguer dans les collections

Evidemment, le système fonctionne de la même manière qu'expliqué ci-dessus, à ceci près que derrière la variable `a`, on trouve la variable `cl` qui spécifie l'endroit où l'on se trouve dans la collection. Il s'agit d'une structure hiérarchique comme nous pouvons le voir sur la Figure 5.12 (regardez également la barre d'URL). Par exemple, si `cl` reçoit la valeur `CL2`, il s'agit de la classification du deuxième type (i.e. *subjects*). Si `cl=CL2.1` alors nous sommes dans une sous-collection (il s'agit du premier *bookshelf* de la liste); si `cl=CL2.1.2` nous sommes dans une sous-sous-collection (il s'agit du deuxième *bookshelf* de la liste), et ainsi de suite.

FIG. 5.12 – *Le programme «library» pour naviguer dans une collection.*

5.8 Conclusions

Dans ce chapitre nous avons vu plus ou moins en profondeur le programme Greenstone: comment créer des collections, comment rechercher de l'information, et finalement le système du point de vue de son architecture et de son implémentation (en voyant succinctement le programme *library*). Nous avons vu ceci car l'implémentation de GreenstoneToAvantgo dépend de l'implémentation de Greenstone. Les arguments de *library* (que nous avons présentés) nous offrent un moyen simple d'identifier les documents, les sections, etc. Prendre en compte les arguments de *library* va nous permettre d'isoler les fonctions de la bibliothèque digitale qui nous intéresse. En effet, ils nous permettront de générer des documents présents dans une collection, mais également de recréer une structure arborescente de fichiers grâce à la structure de l'OID.

Le système Greenstone comme nous l'avons vu a été conçu pour durer et pour stocker beaucoup d'informations de qualité. C'est pour cela que nous nous sommes focalisés sur l'aspect technique. D'une part cela nous donne une introduction au problème des bibliothèques digitales, et d'autre part il est indispensable d'analyser l'existant afin de construire une application correcte. Sans cette étude nous ne serions pas capables de faire une application qui tienne compte des modifications que Greenstone est susceptible de rencontrer. Mais surtout nous prendrions le risque de faire des erreurs d'implémentation.

Nous nous en sommes pas tenus qu'à l'aspect technique. Nous avons également vu en détail l'aspect recherche. Nous avons voulu vous faire la démonstration que le système de recherche d'informations des bibliothèques digitales de Greenstone sont efficaces. Nous voulions également vous convraincre de la qualité du sytème IR, notamment grâce aux options de recherche. Cela nous permet de faire le lien avec le Chapitre 4 et voir un exemple concret de système IR. C'est une des raisons pour lesquelles nous avons insisté sur le système IR, et sur la façon de juger de sa qualité. Les lecteurs attentifs auront remarqué que lorsque Greenstone retournait les résultats il existait toujours un titre faisant office de résumé, permettant à l'utilisateur de choisir ce qu'il veut lire en premier. Il est clair que ceci est également déterminé par l'ordre des résultats.

Nous ne reviendrons plus sur la qualité du système hypertexte, mais ceux qui ont bonne mémoire, se rappeleront sans difficulté, la vue qu'offraient le WebTwig et le Power Browser. C'etait une vue arborescente de n'importe quel site Web. Vous remarquerez aisément que la structure offerte par la bibliothèque est aussi une vue arborescente. GreenstoneToAvantgo héritera

tout naturellement de cette structure et offrira donc autant d'avantages que les deux systèmes précédents (WebTwig et Power Browser).

Nous rappelons que nous poursuivons un double objectif, offrir un système de navigation efficace et donner accès au pays en voie de développement à l'information des bibliothèques digitales de Greenstone. C'est pour cela que nous avons montré au début du chapitre que New Zealand Digital Library Project a créé un certain nombre de collections humanitaires. Les captures d'écran reprenaient ces collections. L'objectif à court terme est donc de servir ces collections aux pays en voie de développement par le biais des PDAs.

Chapitre 6

GreenstoneToAvantgo

Nous allons vous présenter dans ce chapitre le système GreenstoneToAvantgo. Le principe de GreenstoneToAvantgo est de transformer les pages Web des bibliothèques digitales de Greenstone. Le but est de les mettre dans un format spécifique pour qu'elles puissent être compressées et visualisables sur le navigateur AvantGo[1].

Afin de se resituer parmi les deux différentes approches d'accès à l'Internet vues au Chapitre 2 mentionons qu'AvantGo propose un système où l'on crée un Web parallèle. C'est-à-dire que des sites Web sont expressément conçus pour les PDAs.

Le nom de GreenstoneToAvantgo vient du fait qu'il joue le rôle d'interface entre les bibliothèques digitale et AvantGo.

Nous verrons à la Section 6.1 quelles sont les raisons qui motivent notre système. A la Section 6.2, nous discuterons de la compagnie AvantGo et des possibilités qu'elle offre. A la Section 6.3, nous verrons le système GreenstoneToAvantgo: son rôle et son fonctionnement. Nous examinerons comment entreprendre des recherches sur le PDA à la Section 6.4. Et finalement à la Section 6.5 nous verrons brièvement l'implémentation de GreenstoneToAvantGo.

[1]Nous verrons par la suite quel est le navigateur AvantGo.

6.1 Motivations

6.1.1 Les bibliothèques digitales: quantité et de qualité

L'avantage à priori des exemples de systèmes que nous avons vus dans les chapitres précédents est qu'ils sont capables de transformer n'importe quel site du World Wide Web, alors que le système que nous proposons ne se cantonne qu'à transformer les pages Web des bibilothèques digitales de Greenstone.

A ceci nous répondons que si les sites Web ne montrent pas suffisamment de structure, les systèmes WebTwig et Power Browser seront pris en défaut. Ensuite, nous soulignons le fait qu'une bibliothèque digitale a par définition de l'information de qualité. Et, trouver de l'information sur l'Internet représente des fois un sérieux challenge: on y trouve tout et n'importe quoi.

Selon une étude menée par [Bergman, 2001], il existe deux Web: le deep Web et le surface Web. Le deep Web est en fait toute l'information qui est stockée dans des bases de données, et qui est générée dynamiquement dans des pages Web au moment de la navigation. De part leur nature ces pages Web ne sont pas indexées par les moteurs de recherche: elles pages n'existent pas tant qu'elles n'ont pas été générées. Ce Deep Web:

- représente de l'information publique qui est 400 à 550 fois plus gros que la définition commune du WWW;
- c'est 7.500 terabytes d'informations par rapport à 19 terabyates d'informations dans le surface Web;
- c'est 550 milliards de documents individuel versus 1 milliard pour le surface Web;
- c'est plus de 200.000 sites Web existants;
- contient 60 des plus gros sites Web. Ils contiennent ensemble 750 terabytes d'informations. A eux seuls ils excèdent quarante fois la taille du surface Web;
- c'est 50 % de trafic collectif mensuel en plus;
- c'est la part de l'internet qui croît le plus vite;
- la qualité totale du contenu est 1000 à 2000 fois supérieure au surface Web;
- 95% des site Web sont gratuits d'accès.

De plus, NEC Research Institute a plublié une étude dans *nature* (cité dans [Bergman, 2001]). Cette étude estime que les moteurs de recherches n'indexent pas plus de 16% du surface Web. De plus, étant donné qu'ils ne

sont pas capables d'indexer le Deep Web, ils ne recherchent que dans 0.03% du World Wide Web, et malheureusement pas là où se trouve l'information la plus pertinente. Afin d'illustrer ce phénomène nous pouvons voir sur la Figure 6.1 et la Figure 6.2 (images [Bergman, 2001]), le World Wide Web comme un océan et les bateaux de pêche comme les moteurs de recherches.

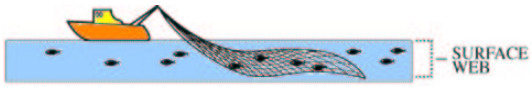

FIG. 6.1 – *Analogie au surface Web.*

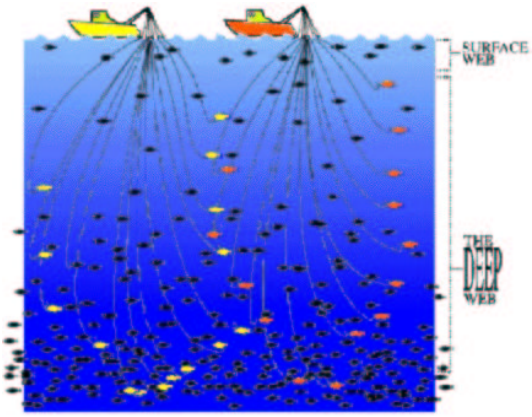

FIG. 6.2 – *Analogie au deep Web.*

Donc pourquoi s'occuper de tous les sites Web, alors que nous savons que la probabilité d'y trouver de l'information intéressante est faible. D'un autre côté Greenstone est gratuit, open-source et de qualité. Il va de soi que plus il se fera connaître plus il y de chance que beaucoup d'universités et autres institutions publiques l'utilisent. Donc en offrant un accès exclusif aux bibliothèques digitales de Greenstone nous effectuons un bon compromis. Nous offrons de l'information de bonne qualité et en grande quantité, et des méthodes de recherche efficaces.

6.1.2 Offrir un accès aux bibliothèques digitales aux pays en voie développement

Les difficultés pour l'accès à l'Internet des pays en voie de développement signifient que, sauf si des actions directes sont prises, l'ère des technologies

de l'information risquerait sans doute de ne pas leur profiter.

Comme nous l'avons déjà signalé, les nouveaux PDAs disposent d'un modem sans fil qui leur permet d'avoir accès à l'Internet (par l'intermédiaire d'opérateurs de téléphonie mobile). Notons également qu'ils sont plus disposés à la recherche d'informations sur l'Internet que les téléphones mobiles. Les raisons étant qu'ils possédent un espace d'affichage plus large et sont d'une puissance supérieure.

De plus, bien que les prix des PDAs soient encore élevés, ils le sont moins que ceux des PCs. De plus, et il n'est pas exclu que leur prix chutent dans les années à venir.

Les nombreux avantages dont bénéficient les PDAs par rapport aux ordinateurs conventionnels et aux téléphones mobiles les privilègient pour l'accès à l'Internet, surtout en Afrique. En effet, au début de l'année 2001, 1 personne sur 60 possédait un téléphone mobile, 1 personne sur 70 possédait un ordinateur et seulement 1 personne sur 80 pouvait avoir accès à l'Internet. La raison est que l'infrastructure pour la téléphonie mobile se développe plus rapidement que l'Internet en Afrique ([ITU TELECOM, 2001]).

Le «Boom» du téléphone mobile

Selon [ITU TELECOM, 2001], on peut véritablement parler d'une explosion de la téléphonie mobile en Afrique. Concrétement, au début de l'année 1991 le nombre d'abonnés au téléphone mobile sur l'ensemble du territoire africain était inférieur à 22 000. Alors qu'au début de l'année 2001, il a été supérieur à 11 millions.

Dans plusieurs pays - comme le Botswana, la Côte d'Ivoire, le Gabon, le Maroc, l'Ouganda, le Rwanda, le Sénégal, les Seychelles, la République d'Afrique du Sud et la Tanzanie - le nombre d'abonnés au téléphone mobile dépasse désormais celui du téléphone fixe. Par contre dans les pays déchirés par la guerre ou par les troubles civils, le développement a été interrompu s'il n'a pas fait marche arrière.

Les raisons du dévéloppement

Toujours selon [ITU TELECOM, 2001], 4 raisons favorisent le développement des téléphones mobiles:

Les services de prépaiement au Sénégal(en 1998) ont incité d'autres pays à faire de même. Ce genre de système permet de faire en sorte que les clients paient à l'avance pour le service fourni. Ils permettent aux clients de contrôler leur consommation et aux opérateurs d'avoir un moyen simple de toucher les redevances. Ces services de prépaiement sont donc bien adaptés aux utilisateurs aux revenus modestes.

En outre, et cela est plus important encore, le désir des opérateurs étrangers d'exploiter les nouveaux débouchés offerts par le téléphone mobile. Ces nouvelles intitiative aident l'Afrique à combler son manque chronique d'investissements dans le développement des télécommunications. Au début de 2001, 18 pays d'Afrique sur les 33 que les Nations Unies ont classés dans la catégorie des pays les moins avancés, avaient déjà octroyé des licences à des opérateurs de téléphones mobiles associant un ou plusieurs investisseurs étrangers.

De plus, la libéralisation généralisée des marchés du mobile dans tout le continent offre des possibilités d'investissements aux opérateurs locaux qui peuvent transposer l'expérience qu'ils ont acquise sur des marchés locaux dans d'autres pays de la région.

Et Finalement, la capacité qu'ont les pouvoirs publics de mobiliser de précieux capitaux en prélevant des redevances au titre de l'octroi de licences pour le téléphone mobile est également en train de doper considérablement les fonds publics dont disposent de nombreux pays pour les télécommunications. Ils créent une demande salutaire portant sur des équipements nouveaux qui devraient permettre de moderniser une infrastructure obsolète et d'étendre la prestation des services à des régions mal desservies.

6.2 AvantGo, Inc

AvantGo, Inc. a été fondé en 1997 et est basé à Hayward (Californie). Le but de cette compagnie est d'adresser les besoins croissants des sociétés, de toutes tailles, de permettre à leurs employés d'être mobiles, ainsi que leurs applications, et ce grâce aux PDAs (Palm OS et Windows CE).

6.2.1 Les services d'AvantGo

AvantGo a développé un grand nombre de services différents (voir `http:\\ avantgo.com`), dont deux nous intéressent. Le premier est le *Mobile Internet Service* (MIS). Il est *gratuit* et permet d'avoir un accès aux *channels*.

Les *channels* sont des sites Web specialement conçus et optimisés pour MIS. Ils traitents de sujets divers tels des news, des horaires d'avions, des cartes routières, la météo, etc. Ils sont conçus et fournis par des entreprises telles Yahoo!, CNN,... et des e-business.

Le second service (i.e.*Custom channel*[2]) qui nous intéresse donne la possibilité, à tout un chacun, de créer son propre *channel* en utilisant les spécifications fournies par AvantGo.

Afin de disposer de ces services, AvantGo fournit gratuitement à ces clients un navigateur Web. Ce navigateur permet aux utilisateurs de visualiser les *channels* qu'ils ont choisis sur le site de AvantGo. La particularité des *channels* est qu'ils sont stockés dans le PDA au moment de la *synchronisation* (ceci sera expliqué à la Section 6.2.2) avec le serveur d'AvantGo. Donc, la navigation dans les *channels* se fait en mode déconnecté.

Si l'utilisateur dispose d'un modem sans fil rien ne l'empêche d'utiliser son navigateur Avantgo pour surfer sur le World Wide Web. Mais dans cas rien n'est fait pour améliorer les pages Web.

6.2.2 Synchronisation

La synchronisation est le processus qui permet de charger le PDA avec les *channels* sélectionnés par les utilisateurs.

Il existe deux manières de synchroniser le PDA avec les *channels*:

1. si l'utilisateur n'a pas de modem sur son PDA, alors l'exigence minimum est un ordinateur disposant d'une connection à l'Internet. Pour ces utilisateurs, AvantGo fournit gratuitement un composant qui s'installe sur le PC. C'est ce composant qui permettra la synchronisation.
2. ou bien l'utilisateur dispose d'un modem sans fil sur son PDA, dans ce cas un autre composant s'ajoute dans le PDA. C'est se composant qui chargera d'effectuer la synchronisation.

Pour la facilité de l'exposé nous expliquerons la synchronisation dans le cas où l'utilisateur ne possède pas de modem sans fil. De plus nous avons développer l'application GreenstoneToAvantgo sous cette condition (dans les deux cas le principe reste le même).

[2]Ce service est gratuit tant qu'il n'y a pas plus de 8 personnes qui y accèdent. Si une personne est intéressée par la création d'un *channel* qui sera accédé par plus de 8 personnes, alors il existe le «Mobile Marking & Commerce Contact» qui permet aux utilisateurs qui le souhaitent de trouver des informations supplémentaires.

On suppose que le PDA est connecté à l'ordinateur à l'aide d'un câble. Comme nous pouvons le voir sur la Figure 6.3, la synchronisation peut être décrite en 4 étapes:

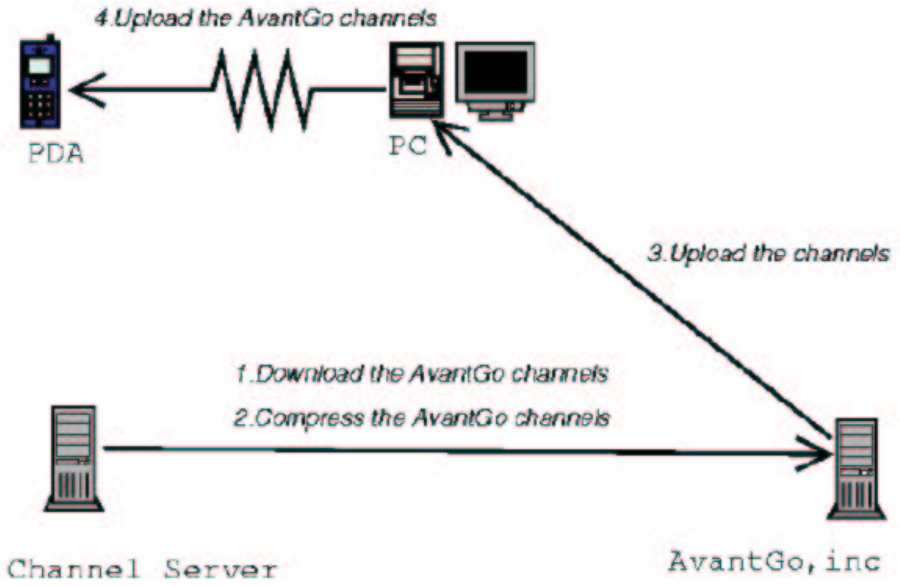

FIG. 6.3 – *Synchronisation entre le serveur Avantgo, le(s) serveur(s) de channels, et le PDA, en utilisant le PC comme intermédiaire.*

1. le serveur détermine le client AvantGo et vérifie la liste des *channels*. Il ouvre une connexion avec chaque serveur de *channels*, et il télécharge en local;
2. Il compresse le contenu des pages avec les technologies d'AvantGo;
3. Il «upload» les *channels* dans le PDA en utilisant le PC (grâce au composant placé à l'intérieur) comme intermédiaire;
4. Les *channels* sont «uploadés» dans le PDA.

Une fois que le processus est terminé l'utilisateur peut visiter les *channels* qu'il a téléchargés, en mode déconnecté.

Remplir un formulaire sur une page Web

AvantGo fournit également une solution pour faire suivre les inputs des utilisateurs aux serveurs qui hébergent les formulaires, remplis off-line. Les inputs

peuvent être, par exemple, les mots clés introduits sur la page d'un moteur de recherches.

Le fonctionnement est le suivant: les inputs sont d'abord stockés dans le PDA, et au moment de la synchronisation ils sont envoyés aux serveurs; ces derniers traitent la requête et renvoient les résultats, qui sont transmis et stockés dans le PDA par l'intermédiaire d'AvantGo. L'utilisateur peut ensuite accéder aux résultats dès qu'il le souhaite.

Le Navigateur d'Avantgo

Comme nous pouvons le voir sur la Figure 6.4, la barre des menus se trouve en haut à droite. Ils contiennent les icônes nécessaires à la navigation, dont les plus importants sont les suivants: flèches gauche et droite et la petite maison qui permet de revernir sur la page d'acceuil pour. Sur la Figure 6.5, on peut voir un exemple de page d'acceuil, où tous les *channels* stockés dans le PDA sont listés. Si le navigateur sélectionne un titre avec son stylus, alors il pourra naviguer à l'intérieur de ce *channel*.

FIG. 6.4 – *Le menu du navigateur AvantGo.*

Les pages AvantGo sont compressées par les technologies AvantGo et conçues à l'aide d'un sous-ensemble de HTML — pour le moment HTML 3.2 pour la version 3.3 d'AvantGo.

6.3 Le rôle de GreenstoneToAvantgo

GreenstoneToAvantGo (GTA) est une interface entre les bibliothèques digitales et AvantGo MIS. Il crée à partir du contenu et de la structure des bibliothèques digitales de Greenstone un *custome channel*. L'idée est de créer automatiquement, «à la volée», un *custome channel* d'une partie de la bibliothèque digitale pour laquelle l'utilisateur a défini une portion: une liste de livres, un livre, une section,...

FIG. 6.5 – *Exemple de home page sur le PDA. Tous les channels sont listés.*

Ce système permet de profiter des avantages du système AvantGo. Outre le navigateur sur PDA, AvantGo fournit également la compression des pages Web et des images. Si l'on ajoute à cela le processus de GTA — il transforme les pages Web en les réduisant à l'essentiel de l'information — alors la quantité d'informations que l'on peut stocker sur le PDA est importante. AvantGo réduit également les images pour qu'elles soient visualisables sur le PDA.

Il autorise l'utilisateur à transférer de 2Mo de données par synchronisation avec un maximum de 20Mo par jour. Les pages de GreenstoneToAvantgo ont une moyenne de 10 à 15 Ko, si on ne prend pas en compte les images. Ce qui veut dire que le nombre maximum de pages par synchronisation varie entre 133 et 200 pages Web (sans compression).

Il permet également de profiter des nombreux avantages qui sont offerts par une bibliothèque digitale.

6.3.1 Processus de création du *custom channel*

Le processus qui va être présenté a été simplifié afin de rendre la discussion élémentaire. Actuellement, il y a encore un certain nombre d'étapes qui sont effectuées manuellement (pour une description complète voir Annexe A).

Interaction AvantGo, la Bibliothèque Digitale et GTA

L'interaction entre la bibliothèque digitale, GTA, AvantGo et le PDA peut être divisée en deux grandes étapes. La première est l'utilisation de GTA (voir Figure 6.6) et la seconde est la synchronisation entre le serveur qui héberge les *custom channels* et AvantGo (voir Figure 6.7). Tout d'abord nous supposons que l'utilisateur:

1. est client de AvantGo (ce qui est gratuit);
2. dispose d'un compte sur un serveur Web afin de pouvoir stocker son *custom channel*. Ce qui permettra à AvantGo de synchroniser le PDA avec le *custom channel*.

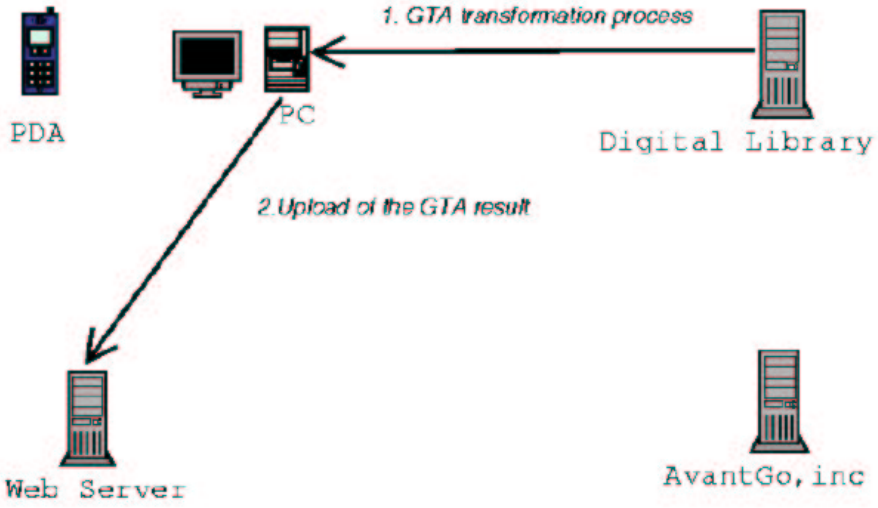

FIG. 6.6 – *Processus de GreenstoneToAvantgo.*

Dès lors le processus peut commencer:

1. quand GTA débute son processus, il télécharge une portion de la bibliothèque digitale (spécifiée par l'utilisateur). Il fait les transformations nécessaires pour créer un *custom channel*— avec le contenu de l'information qui a été sélectionnée. Le résultat est stocké sur le PC sur lequel GTA effectue les traitments;
2. le contenu est transféré sur le serveur Web spécifié par l'utilisateur;

Ensuite, c'est le moment de la synchronisation:

1. le serveur AvantGo détermine l'utilisateur et vérifie la location du *cus-*

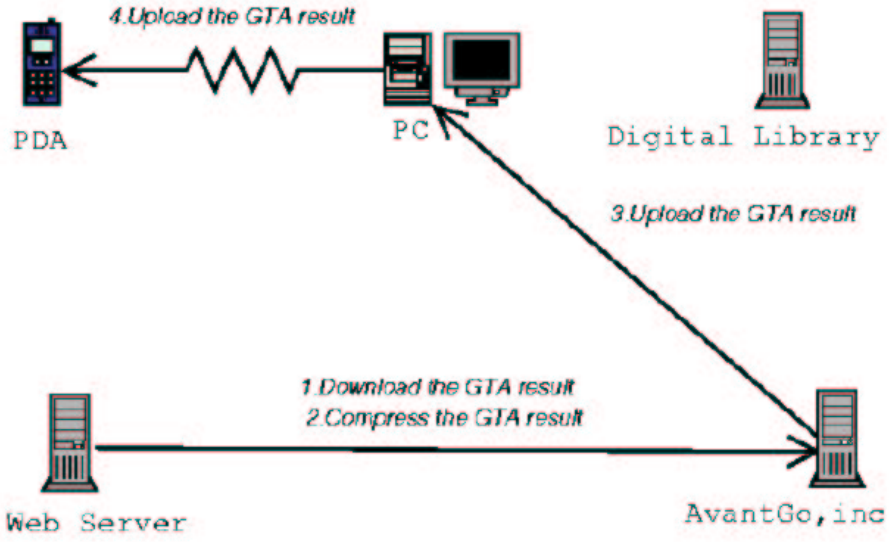

FIG. 6.7 – *La synchronisation du PDA avec le le serveur qui héberge le «GTA custome channel».*

tome channel (i.e. URL sur le serveur Web qui l'hébérge). Ouvre une connexion avec le serveur. AvantGo télécharge le *custom channel GTA*;

2. AvantGo compresse le *custom channel GTA*;
3. le *custom channel GTA* est «uploadé» sur le PC par l'intermédiaire du PC;
4. le *custom channel GTA* est «uploadé» sur le PDA.

Configurer GreenstoneToAvantGo

Avant toute chose, il convient de configurer GTA de manière à ce qu'il soit capable de déterminer la portion de la bibliothèque digitale que l'utilsateur souhaite visualiser sur son PDA (voir Figure 6.8). Une partie d'une collection (e.g. une liste de livre, un livre, une section) ou bien la page Web des résultats d'une requête à la bibliothèque digitale.

On peut voir que dans le premier champ, on peut spécifier la source d'information que l'on souhaite (e.g un livre, une liste de livres). Le second permet de sélectionner l'endroit où stocker le résultat (sur le disque dur en local). Et finalement, le nombre de liens qu'il faut suivre partant de la première page, la taille maximum en Ko du résultat, et l'inclusion des images.

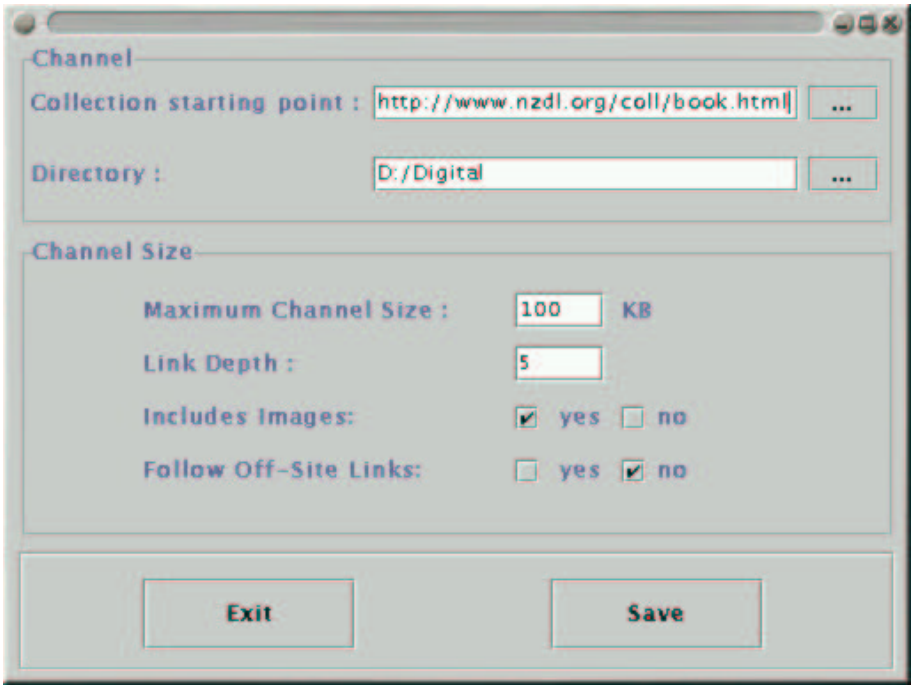

FIG. 6.8 – *Partie de l'interface utilisateur permettant de spécifier les inputs de GTA.*

Remarques:

1. la dernière possibilité, c'est-à-dire suivre les liens extérieurs, n'est pas encore implémentée.
2. l'envoi du *custom channel GTA* vers le serveur n'est pas visible ici. On utilise simplement le protocol *ftp* (file transfer protocol).

6.4 Recherche d'informations en mode déconnecté

6.4.1 Principes

A l'image du Knowledge Agent Bases que nous avons vu à la Section 3.3, la recherche se fait en mode déconnecté. De plus, la structure du sytème hypertexte des *GTA customs channels* est calquée sur celle des bibliothèques digitales de Greentsone.

6.4.2 Recherche par mots clés

La recherche par mots clés se fait à deux niveaux:

1. *le premier niveau* se fait dans la bibliothèque digitale. L'utilisateur en face de son ordinateur fait une première recherche dans la bibliothèque digitale de Greenstone. Cette première recherche lui permet de sélectionner la partie de la bibliothèque qu'il désire télécharger sur son PDA, et consulter plus tard en mode déconnecté. Après cette première recherche il peut configure GreenstonToAvantGo avec les nouveaux paramètres. Et quand il le désire synchroniser son PDA avec son ou ses nouveaux *channel(s)*.
2. *le second niveau* se fait dans les pages présentes sur le PDA. AvantGo offre la possibilité de recherche d'informations dans les *channels* du PDA (voir Figure 6.9).

6.4.3 Navigation

Pour la navigation, on maintient la même structure que les bibliothèques digitales, mais on simplifie la page Web. On peut voir sur la Figure 6.10,

FIG. 6.9 – *Recherche dans les channels présents dans le PDA.*

comment on représente une liste de livres, et sur la Figure 6.11 un livre: l'image a été remplacée par un lien hypertexte[3]. La table des matières n'est montrée que pour la première page. Lorsque l'utilisateur veut revenir en arrière il peut se servir des flèches disponible sur son navigateur.

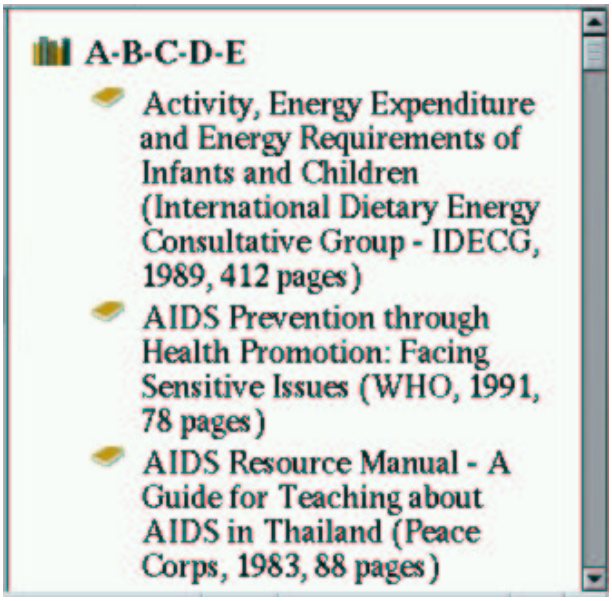

FIG. 6.10 – *Page d'une bibliothèque digitale Greenstone, représentant une liste de livres, qui a été transformée par GreenstoneToAvantgo.*

[3]Nous avons simulé l'affichage du PDA en réduisant la fenêtre d'un navigateur conventionnel.

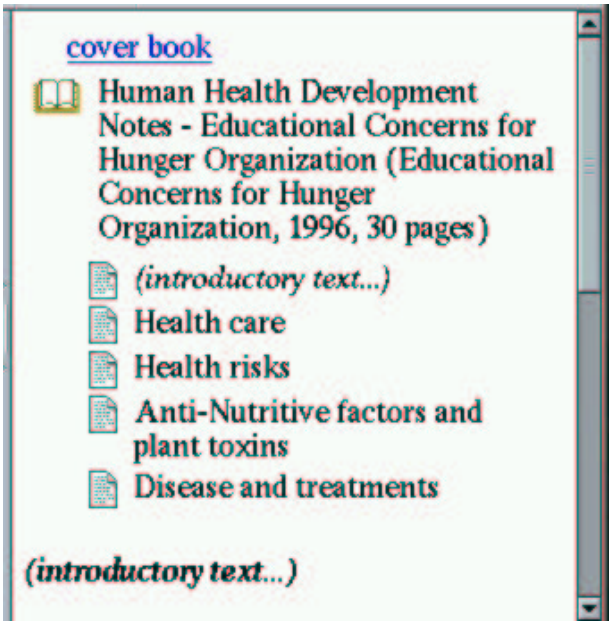

FIG. 6.11 – *Page d'une bibliothèque digitale Greenstone, représentant un livre et sa table des matières, qui a été transformée par GreenstoneToAvantgo.*

6.5 Implémentation

Le choix du langage d'implémentations s'est tout naturellement porté sur Java. Les raisons sont diverses. Java:

- est le standard actuel du marché. La plupart des développeurs le connaissent. Ceci facilite grandement les choses si dans l'avenir l'application doit être étendue;
- supporte l'Unicode. Ceci est fondamental dans l'environnement multilinguistique des bibliothèques digitales;
- est portable sur n'importe quelle plate-forme lorsque la machine virtuelle de Java est installé;
- est extensible. C'est un langage complet qui permet d'étendre toute application;
- est très avantageux pour tous les accès aux réseaux d'Internet. Il offre de nombreuses facilités.

Le travail d'implémentation a nécessité la création de quatres classe principales:

1. *ProjectControler* qui s'occupe de gérer le projet[4]:
   (a) initialiser le projet;
   (b) vérifier la taille du projet en cours;
   (c) vérifier la profondeur du site que l'on traite;
   (d) vérifier si l'on doit prendre en compte les images;
   (e) stopper le projet;
2. *GetWebFile* s'occupe de télécharger les pages Web des bibliothèques digitales;
3. *CgiBinLibrary* permet de créer la structure d'un site Web (plus exactement d'un *custom channel GTA*). Cette partie s'occupe également de parser les arguments CGI du programme *library* (vue à la Section 5.7.2).
4. *WebFileFactory* a pour tâche de simplifier les pages Web.

Nous présentons en Annexe B l'architecture relative à notre application.

6.5.1 *GreenstoneToAvantgo Runtime system*

Avant d'expliquer la séquence des opérations de GTA, nous allons définir quelques notions.

Représentation d'un système hypertexte

Nous pouvons voir un sytème de liens hypertextes comme un arbre où chaque noeud est une page Web, et un lien entre chaque noeud est un lien hypertexte entre deux pages Web (voir Figure 6.12). Supposons que la page que nous

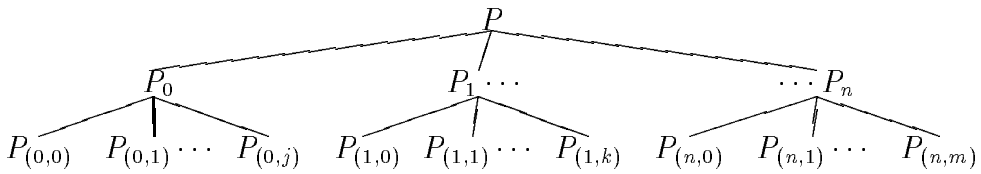

FIG. 6.12 – *Sous-arbre répresentant une partie du système hypertexte.*

sommes en train d'analyser P est un sous-arbre de niveau i; nous l'appelerons

[4]Par projet nous entendons la création d'un *GTA custom channel*

P^i. Les n pages qui sont référencées par P^i sont des pages de niveau $i+1$; nous appelerons ces pages P^{i+1}. De la même sorte, les pages référencées par les pages P^{i+1} sont des pages de niveau $i+2$; nous appelerons ces pages P^{i+2}. Bien entendu, P^0 est la racine de l'arbre. Donc:

$$\begin{aligned} P^i &= \{P\}, \\ P^{i+1} &= \{P_0, P_1, \cdots, P_n\}, \\ P^{i+2} &= \{P_{(0,0)}, P_{(0,1)}, \cdots, P_{(0,j)}, P_{(1,0)}, P_{(1,1)}, \cdots, P_{(1,k)}, P_{(n,0)}, P_{(n,1)}, \cdots, P_{(n,m)}\}. \end{aligned}$$

Un fichier HTML de Greenstone

Lorsque nous parlons des pages Web dans Greenstone, nous considérons les pages qui sont générées dynamiquement par le programme *library*. Un fichier est composé d'un ensemble de lignes. Ces lignes contiennent:

- des balises HTML;
- des références vers d'autres documents;
- des références vers des images;
- du texte;
- et autres.

Un lien référençant un nouveau document est formé d'un appel au program *library* et des arguments nécessaires pour que la page référencée puisse être générée (voir Figure 6.13).

```
<tr valign=top>
 <td valign=top><a href="/gsdl/cgi-bin/library?a=d&cl=CL2.1&d=HASH92b3e33d4cd4d16e000bbb.1">
<img src="/gsdl/images/itext.gif"></a></td><td>Acknowledgments</td>
</tr>
```

FIG. 6.13 – *Exemple d'un lien hypertexte.*

L'un des rôles de la classe *CgiBinLibrary* est de parser la page HTML et d'en extraire les URLs correspondant aux documents. Il doit également renommer le fichier avec un nom approprié à son nouvel emplacement; étant donné que le fichier est téléchargé en local[5]. Par conséquent, il convient également de référencer les fichiers en fonction de leur nouveau nom (voir Figure 6.14).

[5]sur le PC de l'utilisateur où fonctionne GTA.

```
<tr valign=top>
 <td valign=top><a href="CL2/1/HASH92b3e33d4cd4d16e000bbb/1.html">
<img src="/gsdl/images/itext.gif"></a></td><td>Acknowledgments</td>
</tr>
```

FIG. 6.14 – *Exemple d'un lien hypertexte transformé.*

Un soin particulier a été pris pour référencer les fichiers par leur adresse relative[6]. La raison est que le *custom channel GTA* doit être transféré sur un serveur Web. La seule manière d'assurer que les fichiers seront toujours correctement référencés est d'utiliser des adresses relatives.

Séquence des opérations

L'ensemble des opérations peut se diviser en deux phases. La première, le *crawling* a pour but de télécharger les pages Web des bibliothèques digitales. La seconde, a pour but de *transformer* des pages Web précédemment téléchargées.

La méthode utilisée pour télécharger les pages Web est en *largeur d'abord*. Si nous reprenons la représentation d'arbre d'un système hypertexte (voir Figure 6.12), il s'agit de télécharger d'abord les pages Web P^i, puis les pages P^{i+1}, puis les pages P^{i+2}, etc, jusqu'aux feuilles de l'arbre. Pour chaque niveau nous commençons par télécharger les pages les plus à gauche (par exemple pour l'ensemble P^{i+1}, nous commençons par la page P_0, puis P_1, etc, jusqu'à la page P_n). L'ordre des pages à télécharger est mémorisé dans la liste des URLs relative aux pages Web. Nous représentons cette liste par un *Vector* JAVA (voir Figure 6.15).

Pour la transformation des pages Web, nous avons parsé les pages HTML et grâce à certaines heuristiques nous pouvons déterminer le type de la page Web (e.g document, liste de livres). Par exemple, si le lien URL d'un document est associé avec l'icône d'un *book* nous pouvons imaginer que le document aura la structure d'un livre (e.g voir la Figure 5.3 de la page 71). Bien que ce système soit intéressant, dans la pratique tous les documents n'ayant pas une structure identique, nous avons préféré identifier le type des pages Web

[6]Nous rappelons qu'un fichier peut être référencé soit par son adresse absolue dans le système (par exemple, `/home/gta-user/public_html/dl/CL2/1/HASHH92b/file.html`), soit par son adresse relative par exemple si notre répertoire courant est `/home/gta-user/public_html/dl/`, le nom relatif de notre document est `CL2/1/HASHH92b/file.html`

url	url_0	url_1	...	url_n	$url_{(0,0)}$	$url_{(0,1)}$	...	$url_{(0,j)}$
$url_{(1,0)}$	$url_{(1,1)}$	...	$url_{(1,k)}$	$url_{(n,0)}$	$url_{(n,1)}$	...	$url_{(n,m)}$	

FIG. 6.15 – *Réprésentation du sous-arbre de la Figure 6.12 dans un Vector Java.*

en fonction de leur structure hypertexte. Principalement, nous avons regardé la structure des `tables`[7], et la façon dont elles sont agencées.

La séquence des tâches est la suivante:

1. *GetWebFile* télécharge la page Web P;
2. *CgiBinLibrary* reçoit le page Web P, ensuite:
   (a) il extrait les URLs spécifiées dans la page P relatives aux pages Web P^{i+1}. Pour ce faire, il parse P et détermine les pages P^{i+1} en analysant les arguments du programme *library* (voir Figure 6.13);
   (b) il stocke les nouvelles URLs relatives aux pages P^{i+1} dans un *Vector* (voir Figure 6.15);
   (c) il rennome les pages Web de P^{i+1}, recrée une structure de fichier (avec des répertoires, des sous-répertoires, etc);
   (d) Et finalement, il change les URLs de la page P relatives aux pages P^{i+1}afin de les référencer correctement.
3. *WebFileFactory* reçoit la page Web P qu'il transforme. Deux exemples ont été présentés aux Figures 6.10 et 6.11 de la page 100.

Le processus se répéte jusqu'à ce que l'une ou l'autre de ces conditions soient vérifiées:

- la taille maximum du projet soit atteint;
- la profondeur des liens hypertextes est atteint;
- la fin du *Vector* soit atteint.

6.5.2 Evaluation

Le programme est capable de transformer d'autres pages Web que celles qui ont été spécifiées dans le chapitre 5: les collections humanitaires. La

[7]Les pages Web sont conçues grâce à un langage de balises que l'on appele HTML (HyperText Markup Language). Les `tables` sont des balises spéciales qui permettent de structurer la présentation des données sur une page Web

plupart des tests ont été effectués sur les bibliothèques digitales référencées au site de Greenstone (`http://www.greenstone.org`). Ils ont été effectués avec un Palm pilot comme celui présenté dans le Chapitre 1. Vous trouverez en Annexe C le listing des classes présentées.

6.6 Conclusion

GreenstoneToAvantgo en servant d'interface entre les bibliothèques digitales de Greenstone et AvantGo offre une solution complète de navigation et de recherche par mots clés. D'ailleur, comme nous l'avons souligné à maintes reprises, l'information qu'offre les bibliothèques digitales est de qualité. Grâce à cette caractéristique nous réduisons la probabilité de gaspiller les ressources du PDA pour la visualisation de données non pertinentes.

Les documents fournis dans les bibliothèques digitales sont de bonne qualité. Ceci implique que les documents ajoutés présentent une structure cohérente, un titre descriptif, une introduction voire un résumé.On pourrait dans l'avenir profiter des avantages des résumés automatiques au cas où l'utilisateur souhaiterait voir sur son PDA une grande liste de livres et quelques extraits. Ceci exigerait évidemment un plus gros effort d'implémentation. Quoi qu'il en soit, nous pouvons noté que chaque titre est un déjà un petit résumé en soit.

GreenstoneToAvantgo n'offre malheureusement pas de système de proposition automatique de mots. Mais toutefois, tous les avantages précédemment cités rendent ce problème moins sensible. En effet, une recherche de premier niveau se fait préalablement sur un ordinateur conventionnel dans la bibliothèque digitale de Greenstone. Grâce à cette première recherche l'utilisateur réduit l'information qui ne fait pas partie de ses recherches. De plus, le temps de réponse pour la recherche est réduit (rappelons ques les *customs channels GTA* sont stockés sur le PDA).

Nous pouvons imaginer beaucoup de situations d'utilisation, même dans les pays en voie de développement. Un ordinateur connecté à l'Internet reste néammoins une exigence minimum pour permettre à GreenstoneToAvantgo de fonctionner. Cette exigence est tout à fait envisageable dans les pays en voie de développement puisque l'on trouve en Afrique beaucoup de lieux publics disposant d'ordinateur, tels que les universités.

L'étudiant par exemple peut selectionner 6 livres et créer 6 *customs channels GTA* (même si la somme des livres est supérieure à la capacité du PDA).

Quand il a besoin d'un de ces livres, il synchronise son PDA avec les *customs channels GTA* grâce au réseau mobile. Il peut alors lire son livre à son aise. Lorsqu'il en a terminé la lecture, il le retire de son PDA pour fournir de l'espace disponnible afin de synchroniser «le second livre», et ainsi de suite jusqu'à ce qu'il ait lu l'ensemble des livres.

S'il le désire, il peut relire les livres précédents étant donné qu'ils sont sur le serveur Web (tant qu'il ne les a pas expréssement supprimés).

Conclusion

Au terme de cette étude nous sommes en mesure de répondre à l'objectif de ce mémoire. Nous rappelons que nous souhaitions évaluer la solution offerte par GreenstoneToAvantgo. Afin d'évaluer la solution nous allons répondre aux deux questions relatives aux problématiques posées à l'introduction.

GreenstoneToAvantgo répond aux exigences techniques. En effet, le mode de recherche en mode déconnecté permet de pallier les limitations de la bande passante. Les pages sont donc chargées plus rapidement sur le navigateur du PDA. Les pages HTML sont simplifiées à l'extrême, le navigateur n'a donc aucun problème à montrer les pages Web. De plus, les technologies d'AvantGo jouent le rôle de proxy en compressant les pages Web; ceci a pour conséquence d'augmenter le nombre de pages que l'on peut stocker sur le PDA. Les technologies d'AvantGo réduisent également la taille des images pour qu'elles soient dans une bonne taille pour être visualisées sur les PDAs.

GreenstoneToAvantgo répond aux exigences relatives à l'aspect humain. En effet, il remet les pages en forme pour qu'elles soient adaptées à l'affichage réduit des PDAs. De plus il tient compte des comportements des utilisateurs dans la recherche d'informations.

Le mode de recherche est complet. Il est possible de faire une recherche par mots clés dans la bibliothèque digitale; de faire une recherche par mots clés en local sur le PDA grâce au navigateur d'AvantGo. GreenstoneToAvanGo crée un petit site Web (*GTA custom channel* en fait) sur le PDA de l'utilisateur. Il est structuré de manière efficace; il maximise l'efficacité de la navigation de l'utilisateur. Comme nous l'avons montré une structure arborescente avec des titres explicites (voire des résumés) facilite grandement la navigation et la découverte d'informations pertinentes.

GreenstoneToAvantgo hérite de tous les avantages d'une recherche sur une bibliothèque digitale. Il dispose de tous les avantages offerts par les technologies d'AvantGo. Nous disposons d'un système qui n'a pas à rougir des

systèmes que nous avons présentés précédemment. De plus, il demande peu d'efforts d'implémentation. Nous rappelons que GreenstoneToAvantgo transforme les pages Web des bibliothèques digitales. Ces pages sont en général structurées de manière prévisible. Mais il ne s'agit pas de transformer toutes les pages du World Wide Web.

Pour ce qui est du contenu, comme nous l'avons montré les bibliothèques disposent d'informations de qualité. Il y a donc plus de chance de trouver de l'information pertinente dans une bibliothèque digitale que dans le World Wide Web. De plus, une recherche sur le Web se basant sur des moteurs de recherches conventionnels n'investigue qu'une infime partie du World Wide Web. Or nous savons que les informations de qualité sont stockées dans des bases de données et les pages Web associées à ces informations sont générées dynamiquement par le biais d'une requête. Ce qui a pour conséquence que ces pages n'existent que pour un instant et qu'elles ne sont pas indéxées par les moteurs de recherche conventionnels. De plus comme Greenstone est gratuit et que chaque bibliothèque digitale peut gérer des gygabytes de données, le problème de la quantité d'informations ne devrait pas être un désavantage.

Certains seraient enclin à critiquer l'usage qui est fait d'un système de navigation et de compression commercial tel qu'offre AvantGo. Mais rappelons que GreenstoneToAvantGo est pour le moment encore dans une phase de prototypage. Si le système donne de bon résultats et si nous sommes en mesure de transformer une grande majorité des collections pour qu'elles puissent être transportées sur PDA, alors rien ne nous empêche d'implémenter notre propre navigateur et un proxy qui jouera le même rôle qu'AvantGo.

GreenstoneToAvantgo a-t-il atteint l'objectif d'offrir l'accès aux bibliothèques digitales de Greenstone ? Il nous semble qu'il est un peu trop tôt pour répondre à cette question. Si nous considérons le niveau technique et les principes théoriques présentés, nous pouvons répondre par oui. Mais afin de confirmer ceci il faudrait mettre en place une expérimentation. Nous fournirions à chaque utilisateur un PDA pour une période de temps déterminée afin qu'ils puissent nous donner leurs impressions du sytème «Greenstone / GreenstoneToAvantgo / AvantGo».

Il est évident que la poursuite de l'objectif humanitaire qui est de fournir accès aux bibliothèques digitales de Greenstone aux personnes des pays en voie de développement est sujet aux mêmes remarques faites ci-dessus. Mais, le développement des réseaux mobiles dans les pays en voie de développement, et plus particulièrement en Afrique nous fait penser qu'il n'est pas inutile de développer des solutions du type GreenstoneToAvantgo.

Il se pourrait qu'à l'avenir il soit plus courant d'accéder à l'Internet par le biais des PDAs que par tout autre médium, surtout en Afrique, le prix d'un PDA étant inférieur au prix d'un ordinateur. Ils offrent des fonctionnalités avancées: ce sont de véritables mini-ordinateurs. Ils peuvent accéder au World Wide Web par le biais des réseaux sans fil. Et ils ont deux grands avantages sur les téléphones mobiles: ils sont plus puissants et ont une taille d'affichage plus grande.

Bibliographie

[Aridor et al., 2001] Aridor, Y., Carmel, D., Maarek, Y. S., Soffer, A., and Lempel, R. (2001). Knowledge encapsulation for focused search from pervasive devices. In *World Wide Web*, pages 754–764.

[AvantGo, 2001] AvantGo (2001). *Help Guide for AvantGo*.

[Barbosa, 2001] Barbosa, A. M. (2001). Overview of text summarization in the context of information retrieval and interpretation: Applications for web pages summarization. Research Article for the Audiovisual Institute of the Pompeu Fabra University in the Context of the doctorate program in computer science and digital communication.

[Bederson and Hollan, 1995] Bederson, B. B. and Hollan, J. D. (1995). Pad++: a zoomable graphical interface system. In *Conference companion on Human factors in computing systems*, pages 23–24. ACM Press.

[Berger and Mittal, 2000] Berger, A. L. and Mittal, V. O. (2000). Ocelot: a system for summarizing web pages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 144–151. ACM Press.

[Bergman, 2001] Bergman, M. K. (2001). The deep web: Surfacing hidden value. Technical report, University of Michigan Press.

[Blyaert, 1999] Blyaert, L. (1999). Ce très cher PDA... *Datanews*, 231.

[Buchanan et al., 2001] Buchanan, G., Farran, S., Jones, M., Thimbleby, H., Mardsen, G., and Pazzani, M. (2001). Improving mobile internet usability. In *Proceedings of 10th International World Wide Web Conference*, Hong Kong.

[Buyukkokten et al., 2000a] Buyukkokten, O., Garcia-Molina, H., and Paepcke, A. (2000a). Focused Web searching with PDAs. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):213–230.

[Buyukkokten et al., 2001] Buyukkokten, O., Garcia-Molina, H., and Paepcke, A. (2001). Accordion summarization for end-game browsing on pdas and cellular phones. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 213–220. ACM Press.

[Buyukkokten et al., 2000b] Buyukkokten, O., Garcia-Molina, H., Paepcke, A., and Winograd, T. (2000b). Power browser: Efficient web browsing for PDAs. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 430–437. ACM.

[Edmundson, 1969] Edmundson, H. P. (1969). New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

[Fuhr, 2001] Fuhr, N. (2001). DL and IR: models and methods, metadata and evaluation.

[Hovy and Lin, 1997] Hovy, E. and Lin, C.-Y. (1997). Automated text summarization in summarist.

[ITU TELECOM, 2001] ITU TELECOM (2001). Bringing the digital divide. In *5th Exhibition of ITU TELECOM Africa 2001*, Johannesburg. Place des Nations, CH-1211 Geneva 20, Switzerland.

[Jones et al., 2000] Jones, M., Buchanan, G., Thimbleby, H., and Mardsen, G. (2000). User interfaces for mobile web devices www9 mobile workshop position paper. In *Poster of 9th International World Wide Web Conference*, Amsterdam.

[Jones et al., 1999a] Jones, M., Mardsen, G., Mohd-Nasir, N., Boone, K., and Buchanan, G. (1999a). Improving web interaction on small displays. In *Proceedings of 8th International World Wide Web Conference*, Toronto.

[Jones et al., 1999b] Jones, M., Mardsen, G., Mohd-Nasir, N., and Buchanan, G. (1999b). A sited-based outliner for small screen web access. In *Proceedings of 8th International World Wide Web Conference*, Toronto.

[Kamba et al., 1996] Kamba, T., Elson, S. A., Harpold, T., Stamper, T., and Sukaviriya, P. (1996). Using small screen space more efficiently. In *Conference proceedings on Human factors in computing systems*, pages 383–390. ACM Press.

[Kurtenbach and Buxton, 1994] Kurtenbach, G. and Buxton, W. (1994). User learning and performance with marking menus. In *Conference proceedings on Human factors in computing systems :?celebrating interdependence?*, pages 258–264. ACM Press.

[Lyman and Varian, 2000] Lyman, P. and Varian, H. R. (2000). How much information? Technical report, University of California at Berkley.

[Morkes and Nielsen, 1997] Morkes, J. and Nielsen, J. (1997). Concise, scanable and objective: How to write for the web. `http://www.useit.com/papers/webwriting/wrinting.html`.

[Nielsen, 1999a] Nielsen, J. (1999a). Graceful degration of scalable internet services, wap: wrong approach to portability. `http://www.useit.com/alertbox/991031.html`.

[Nielsen, 1999b] Nielsen, J. (1999b). Predictions for the web 2000. `http://www.useit.com/alertbox/991226.html`.

[Pillou, 2002] Pillou, J.-F. (2002). Présentation de XML.

[Tambe, 2000] Tambe, R. (2000). Zooming user interfaces for hand-held devices.

[Theng et al., 1996a] Theng, Y., Rigny, C., Thimbleby, H., and Jones, M. (1996a). Improved conceptual design for better hypertext.

[Theng et al., 1996b] Theng, Y. L., Jones, M., and Thimbleby, H. (1996b). Lost in hyperspace: Psychological problem or bad design? In *1st Asia Pacific Conference on Computer Human Interaction*, pages 387–396.

[Witten et al., 2000] Witten, I. H., Boddie, S. J., Bainbridge, D., and McNab, R. J. (2000). Greenstone: a comprehensive open-source digital library software system. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 113–121. ACM Press.

[Witten and Boddy, 2001a] Witten, I. H. and Boddy, S. (2001a). *Greenstone Digital Library: Developers's guide.* Departement of Computer Science, University of Waikato, New Zealand.

[Witten and Boddy, 2001b] Witten, I. H. and Boddy, S. (2001b). *Greenstone Digital Library: User's guide.* Departement of Computer Science, University of Waikato, New Zealand.

[Witten et al., 2001] Witten, I. H., Loots, M., Trujillo, M. F., and Bainbridge, D. (2001). The promise of digital libraries in developing countries. *Communications of the ACM*, 44(5):82–85.

Annexe A

User Documentation

If you are a Personal Digital Assistant (PDA) user who would like to access a *digital library* (DL), it is possible by using *GreenstoneToAvantgo* (GTA) software. This software is part of the **New Zealand Digital Library** (NZDL) project led at the **University of Waikato** [Witten et al., 2000]. It is open-source software, issued under the terms of the GNU General Public License.

In this user documentation, we will first describe what the NZDL is. We will then have a look at AvantGo services [AvantGo, 2001]. Finally we will see precisely the steps to follow to use GTA.

A.1 New Zealand Digital Library Project

NZDL project has provided the Greenstone software in collaboration with UNESCO[1] and the Human Info NGO[2]([Witten and Boddy, 2001b]). It is a suite of software for building and distributing DL collections[3]. It provides a new way of organising information and publishing it on the Internet or on CD-ROM.

The aim of the software is to empower users, particularly in universities, libraries, and other public service institutions, to build their own DL. DL are

[1] `http://www.unesco.org`
[2] `http://humaninfo.org`
[3] An example at `http://www.nzdl.org`

radically reforming how information is disseminated and acquired in UNESCO's partner communities and institutions in the fields of education, science and culture around the world, and particularly in developing countries. The hope is that this software will encourage the effective deployment of digital libraries to share information and place it in the public domain.

A.2 AvantGo,Inc

AvantGo,Inc. is a company founded in 1997 and based in Hayward (California). The goal of this company is to address the growing need of companies of all size for allowing their workforce to be mobile by the use of wireless devices such as Palm OS, Pocket PC and Windows CE.

A.2.1 AvantGo services

AvantGo has developed a lot of different services [AvantGo, 2001], but the most interesting for PDA users who desire to have access to the Internet is the *Mobile Internet Service* (MIS).It is a free service which provides an avantgo-client software.It gives a way for accessing Web pages from the World Wide Web (WWW) and avantgo channels : Web sites specially designed and optimised for MIS and built with avantgo pages. Channels include news,stock quotes,flight schedules, movie listings, restaurant reviews, maps, weather and much more from brand-name content providers, as Yahoo!,CNN,...and e-business.

Create a Customer Channel is another important service which gives the possibility to create a personalised channel using the specification provided[4]. This is relevant to understand how to use GTA: it will create a new channel with the content of a digital library built by Greenstone software.

[4]This service is free for up to eight subscribers per web domain. If someone interested in creating a channel that will be accessed by more than eight subscribers, or if the existing channel grows to more than eight subscribers Mobile Marketing & Commerce Contact Form at `http://avantgo.com/products/businesses/marketing_commerce/forms/markcomm_contact.html` allows customers to find out about options for expanding their Custom Channel service

Unlike the use of MIS to access the WWW, which is done on-line, the visualisation of the channel is done off-line: you can browse all the channels without being connected to the Internet. All the channels are stored in the PDA at the *synchronisation* time with avantgo server, this will be explained in Section A.2.2. When we talk about off-line one idea which comes straightforward to mind is: How to fill a form? AvantGo provides a way to do it, allowing you to fill the form off-line (such as query to a data base) and at the next synchronisation, all data will be sent, and results will be stored back in your PDA.

Avantgo-client is installed both on your PC and on your PDA. One component is presented inside the PDA as a standard Navigator such as Netscape Navigator or Internet Explorer, with navigation bar, scroll bar,etc. and the other one is presented in the computer as an interface between avantgo server and your PDA. Avantgo pages are compressed by AvantGo technologies and designed with a subset of HTML – currently HTML 3.2 for AvantGo version 3.3 –, addressing the requirements concerning PDA as e.g. small screen, low memory,...and that's why a Web server is able to store the channels.

A.2.2 Synchronisation

As soon you have become an AvantGo member and have installed the avantgo-client software for MIS, you will be able to *subscribe* the channels you would like to consult later on your PDA.

The term used to specify the way in which the PDA is uploaded with the selected avantgo channels is called *synchronisation*. As shown in Figure A.1, the synchronisation can be described in 4 steps. We assume this user has already subscribed the desired channels at the AvantGo Web site, and that the PDA is ready to be connected with the PC.

1. Avantgo server determines the avantgo-client and checks the list of channels.It opens a connection with each of these channels host server, and downloads them locally;
2. It compresses the content with AvantGo technologies;
3. It uploads the channels into the PDA using the PC as intermediary;
4. Channels are uploaded into the PDA.

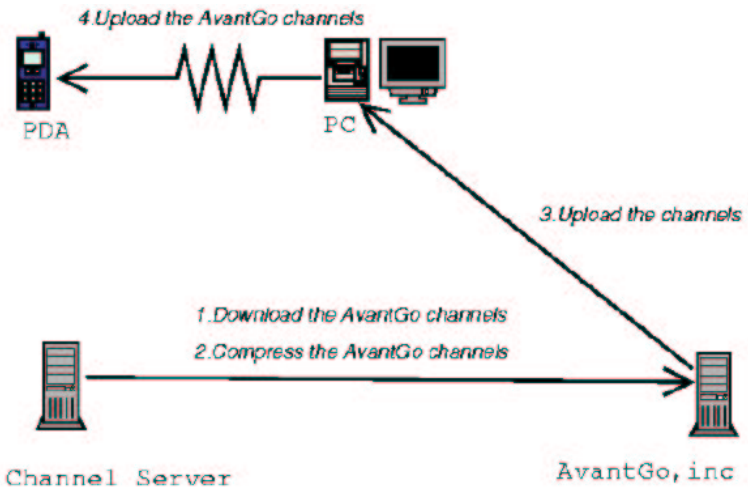

FIG. A.1 – *Synchronisation.*

Once the process is done it disconnects with the channels host servers and the avantgo-client. The user is now able to check every avantgo channel off-line in any place at any time.

A.3 GreenstoneToAvantgo software

GTA software creates from greenstone digital library a Web site specially designed for Avantgo MIS. The idea is to create on the fly a new personalised channel from a selected part of greenstone DL.

I will first explain the interaction between GTA, the greenstone DL and AvantGo. Because it's just a prototype the use is still a little tricky. Further, in the next sections I will present precisely the different steps we had to follow to configure Avantgo and GTA.

A.3.1 Interaction between GTA, greenstone DL and AvantGo

This interaction can be divided in two big steps as shown in Figure A.2 and Figure A.3. First step as the use of GTA and second one as the synchronisation with the avantgo server as explained at Section A.2.2.

First of all, we assume that you are an AvantGo member (see Section A.3.2) and that you already have configured AvantGo by creating your personal channel (see Section A.3.2). We also assume that you have configured GTA in order to determine part of greenstone DL you would like to convert into an avantgo channel (see Section A.3.3).

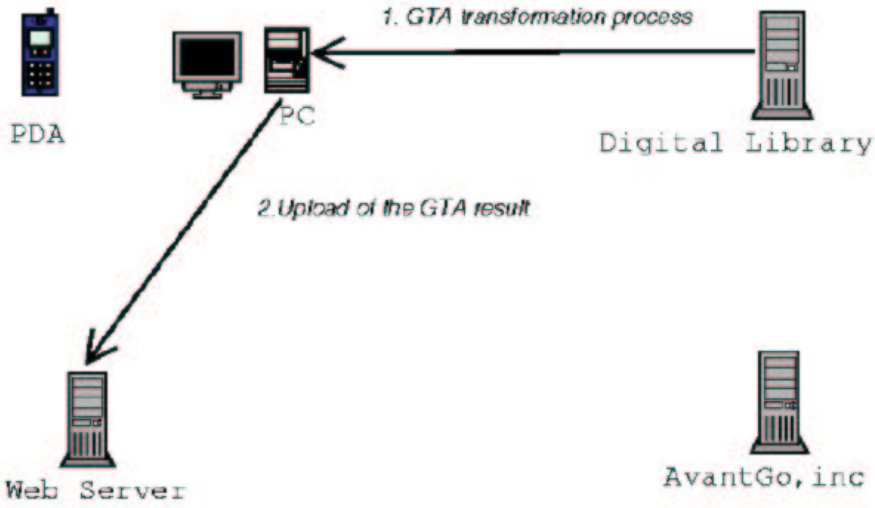

FIG. A.2 – *GreenstoneToAvantGo.*

1. When the GTA runs, it downloads a part of greenstone DL and converts it into an avantgo channel; the result is stored locally in your PC;
2. You transfer by yourself the GTA result (channel) into a Web Server.

And now it's time for synchronisation as shown in Figure A.3.

1. Avantgo server determines the avantgo-client and checks the location of the channels, opens a connection with each of these channels host server, and downloads them locally;
2. Avantgo server compresses the channels with AvantGo technologies;
3. It uploads the channels into the PDA using the PC as intermediary;
4. Channels are uploaded into the PDA.

The process is done, you just have to look at the result in your PDA. To simplify the process GTA has to:

1. Automatically configure AvantGo in order to create the custom channel.

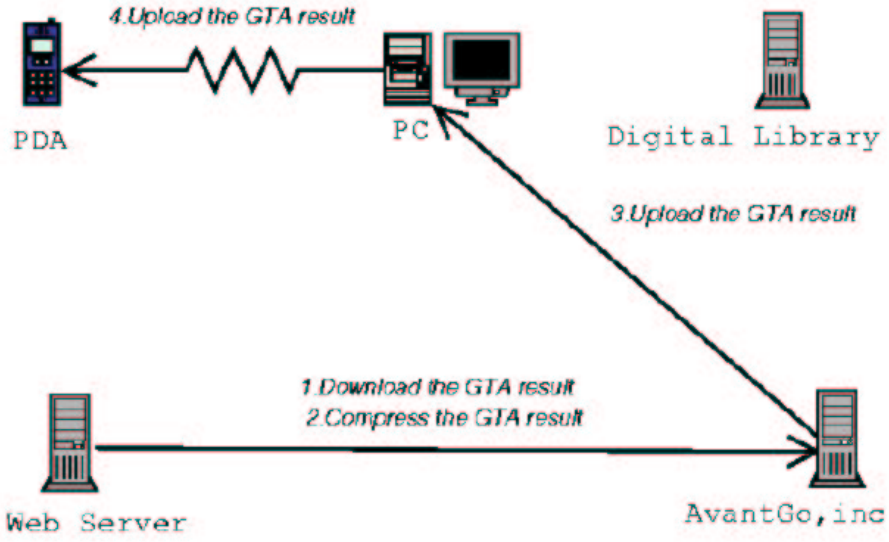

FIG. A.3 – *AvantGo synchronisation.*

2. Automatically, upload the channels into a selected Web server when it has finished the avantgo pages conversion.

In such a manner, the user would only have to run GTA with the specific parameters (DL location, DL channel location on its Web server and finally the login and password at Avantgo) and synchronise as usual his PDA with AvantGo when the GTA process is finished.

A.3.2 AvantGo server configuration

First of all, when you sign up you'll have to download Mobile Internet Service Software (MIS). The software will be installed partially on your computer and your handheld device. Take care of choosing the appropriate avantgo-client depending on your Operating System and your PDA before installation [5].

When this has been done you can freely subscribe to channels provided by Avantgo as The Wall Street Journal, CNN, or Yahoo! and browse them off-line.

[5] `http://avantgo.com/support/mobile_support/downloads.html`

Create your personal channel

AvantGo's technology allows you to create your channel: a custom channel, using existing tools e.g. HTML and provided documentation by AvantGo. That's why a standard Web server is able to store channels.

Configure AvantGo

Once you are logged in the Avantgo Web site you can edit your account and create a channel. You can choose whether to create it via the Create Custom Channel Wizard or the Create Custom Channel Classic as shown in Figure A.4. It will ask you the location of your personal channel; you don't have to give the address of the real Greenstone digital library but the address of the Web server where you stored it. For instance, if you take a book from the `http://www.nzdl.org/book.html` you have to give the address of this book in the Web server where you stored the GTA result e.g. in `http://www.cs.waikato.ac.nz/~Myaccount/MyChannel/book.html`.

This will allow AvantGo to synchronise your PDA with your personalised channel.

A.3.3 GreenstoneToAvantgo configuration

How to launch GreenstoneToAvantgo? On your console you have to type: `java -jar greenstonetoavantgo.jar`

When you launch the application you will have the first window. To start a new project click on menu then click on new. This will open a new window as shown in Figure A.5. It is a form similar to the AvantGo's form you fill in when you want to create a Custom Channel.

The first text field specifies the source location of the information in the Greenstone digital library you would like to access with your PDA, for instance a book or the section of a book. The second text field is the place where you want to store the result on your hard disk. Because GreenstoneToAvantgo is a prototype, it's your own task to transfer the result to a Web server if it is necessary. You can set different preferences concerning the

FIG. A.4 – *Create a Custom Channel after editing your account.*

maximum size of your project, if you'll included images, the number of links you'd like to follow and finally if you want to follow external links (this hasn't been implemented yet).

Save the setting and start the project. It can take a moment depending on the size of your request.

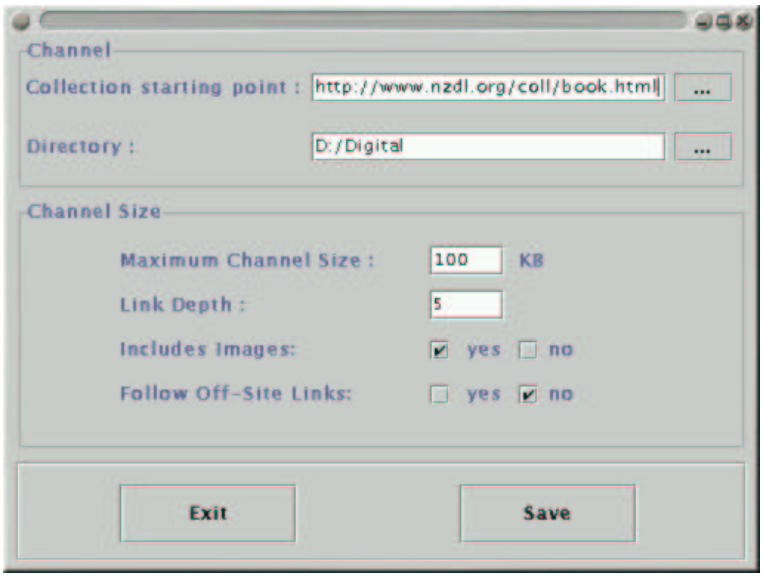

FIG. A.5 – *The form to fill in to save the settings of the channel.*

Annexe B

Architecture of de GreenstoneToAvantGo

B.1 Class Diagram

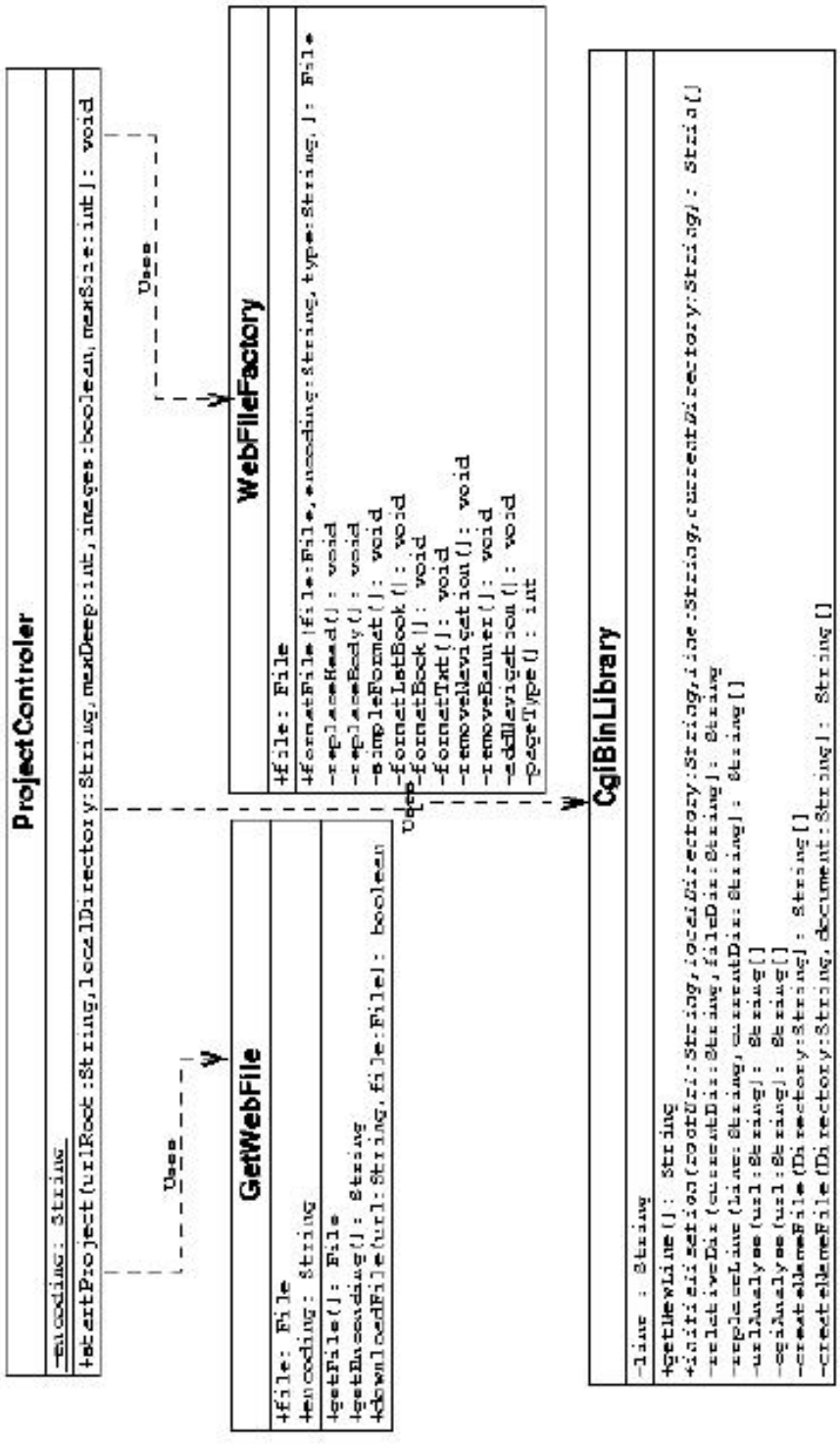

FIG. B.1 – *General Architecture of GreenstoneToAvantgo.*

B.2 Notations

We use the UML notations.

Classes

Illustrate classes with rectangles divided into compartments (Table B.1). The name of the class is placed in the first partition (centered, bolded, and capitalized), list the attributes in the second partition, and write operations into the third.

Class Name
attribute:Type=initialValue
operation(arg list): return typ

TAB. B.1 – *Classes.*

Visibility

We use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class (see Table B.3).

+ *public*
- *private*
♯ *protected*

TAB. B.2 – *Visibility*

An example of how to use it is shown at Table B.3:

Associations

The word «Uses» is simply the name of the association between the two classes. The arrow symbol indicates the sense of the lecture (e.g Project-Controler «Uses» CgiBinLibrary means that the ProjectControler class uses CgiBinLibrary).

Class Name
- attribute:Type=initialValue - attribute
+ operation + operation + operation

TAB. B.3 – *Example of visibility*

Annexe C

Listing des 4 classes les plus importantes

C.1 ProjectContoler

```
package greenstonetoavantgo.core;

import greenstonetoavantgo.*;
import greenstonetoavantgo.chi.*;
import java.io.*;
import java.net.*;
import java.util.*;

/**
 * ProjectControler controls all the process,
 * checking variables such as size
 * of the work which has alreafy been done,
 * the number of links followed
 * until now. Creating the file that close the
 * project (these files has
 * as comment: this is the end of your query).
 * It launchs the others classes;
 * GetWebFile,CgiBinLibrary,WebFileFactory and
 * ImageTreatement.
 *
 */
```

```
public class ProjectControler
{
  private File fl;
  private BufferedReader br;
  private PrintWriter pw;
  private String encoding;
  private MainGuiControler m;

  public ProjectControler(MainGuiControler mn){
    this.pw=null;
    this.br=null;
    this.m = mn;
  }
    /** This is the main method of this class.
     * @param root this is the project starting point
     * @param localDirectory this is the directory
     * storing the project result
     * @param max this is the maximum number of links
     * to follow
     * @param im this is a boolean indicating whether
     * images should be included in the project
     * @param maxsize this is the maximum project size
     */
  public void startProject(String root, String localDirectory,
   int max, boolean im, long maxsize){

    try{
      m.setComment(localDirectory+" "+max+" "+
   im+" "+maxsize+" ");
    }catch(Exception ere){}
    max=max+2;
    int[] depth=new int[max];
    int i = 0;
    depth[0]=0;
    depth[1]=0;
    boolean stop=false;
    long size=0;
    boolean sizestop=false;
    /*
      source is the url of the next file to download
```

```
      location is this file current directory
      directory this is the directory we may to build
      typage is the type of page, this will be
      revelant in formating part
    */
    Vector source=new Vector();
    Vector location=new Vector();
    Vector directory=new  Vector();
    Vector typepage=new Vector();

    /*
        Create all the object that we'll be needed.
        GetWebFile download a web file
        WebFileFactory formats this web file
        CgiBinLibrary to analyse the cgi embbed
in this file. ImageTreatement for the
download of the image.
    */
    GetWebFile GWF=new GetWebFile(m.getMain().getLogin(),
      m.getMain().getPass(),m.getMain().getCach());
    WebFileFactory FWF=new WebFileFactory();
    CgiBinLibrary CBL=new CgiBinLibrary();
    ImageTreatment IT=new ImageTreatment();

    String[] lt=CBL.initialisation(root,localDirectory,
       m.getMain().getLogin(),m.getMain().getPass(),
       m.getMain().getCach());
    String link=lt[0];

    String remoteAbsUrl=lt[3];
    String url=lt[2];
    String dir=localDirectory+lt[0];
    File pathdir=new File(dir);
    /* We create the new file which'll be the copy of the
       remote file (the webfile actually)
    */
    File fl=new File(localDirectory+lt[0]+
     java.io.File.separatorChar+lt[1]);

    try{
    pathdir.mkdirs();
```

```
    fl.createNewFile();
    //  index.createNewFile();
    }catch(IOException e){System.out.println(
"Error in Retry during initialisation :"+ e);}

    /* This is for the initialisation */
    source.add(url);
    location.add(fl);
    directory.add(dir);
    typepage.add("index");

    /* The source contains all the url that we find in
       the webfile
    */
    while((!source.isEmpty())&&(!(stop))&&(!(sizestop))){
      dir=(String)directory.remove(0);
      fl=(File)location.remove(0);
      url=(String)source.remove(0);
      String type=(String)typepage.remove(0);
      /************************GetWebFile***************/
      GWF.work(url,fl);
      fl=GWF.getFile();
      encoding=GWF.getEncoding();
      /**************page formatting******************/
      fl=FWF.formatFile(fl,encoding,type);
      /**********************Formating of the urls******/
      try{
File tmp=createFileStream(fl,encoding);
String theLine;
while((theLine=this.br.readLine())!=null){
    /* We replace the line.*/
  lt=CBL.replace(theLine,dir.substring(
      localDirectory.length()));
  size+=IT.getImage(remoteAbsUrl,localDirectory,
    CBL.getNewLine(),dir.substring(
localDirectory.length()),im);
  /* Here we print in the file created the new
     line formated in ImageTreatement.
  */
  this.pw.println(IT.getNewLine());
  if(lt!=null){
```

```
      /* Here we are looking if the new file that
 we want to download doesn't go out. Actually, in
 the download and the formating of
 the collection we want the download going
 in deep, and not going up.
      */
    if(lt[0].startsWith(link)){
/* We create the new directory of the new file
   with the localDirectory and the relative
   directory that has been build in CgiBinLibrary.
   We do that because for creating the
   file, we need the whole name.*/
      pathdir=new File(localDirectory+lt[0]);
      pathdir.mkdirs();
      /* It's the same than before but for the file. */
      File fe=new File(localDirectory+lt[0]+
       java.io.File.separatorChar+lt[1]);
      if(fe.createNewFile()){
source.add(remoteAbsUrl+'/'+lt[2]);
location.add(fe);
directory.add(localDirectory+lt[0]);
typepage.add(IT.getFormatType()+CBL.getFrame());
depth[i+1]++;
      }
    }
  }
}
boolean b=tmp.delete();
System.out.println("Pro "+b);
this.pw.close();
      }
      catch(IOException er){System.out.println(
  "Error in Retry :"+er);}

      size+=fl.length();
      if(lt!=null ){
      }
      if(size>maxsize){
sizestop=true;
      }
```

```
      if (depth[i]==0){
i++;
if(i>=(max-1)){
  stop=true;
}
else{
  depth[i+1]=0;
}
      }
      else{
depth[i]--;
      }
      try{
      m.setComment(" FILE: "+
   this.fl.getAbsolutePath()+" TOTAL SIZE: ");
      m.setComment(size+" Bytes.");}
      catch(Exception erre){}
      // }
    }//end of general while
    closeProject(directory,location,typepage,source);
  }//end of startDownload

  private File createFileStream(File file,String encoding)
      throws IOException{
    String fname=file.getAbsolutePath();
    File tmp=new File(fname+".temp");
    file.renameTo(tmp);
    //file.delete();
    delete(file);
    //System.out.println(tmp.getAbsolutePath());
    this.fl=new File(fname);
    this.fl.createNewFile();
    //System.out.println(this.fl.getAbsolutePath());

    FileOutputStream fos = new FileOutputStream(this.fl);
    Writer w=new BufferedWriter(
new OutputStreamWriter(fos,encoding));
    /*PrintWriter*/this.pw=new PrintWriter(w);

    FileInputStream fis = new FileInputStream(tmp);
    /* BufferedReader*/ this.br=new BufferedReader(
```

```
new InputStreamReader(fis,encoding));

    return tmp;
  }
  private void stopDownload(){
  }
  /**
   * Close the project by put as comment: "This is
   * the end of your query" in
   * all the file that have been create and so
   * referenced but not downloaded
   * @param Vector directory
   * @param Vector location
   * @param Vector typage
   * @param Vector source
   */
  private void closeProject(Vector directory,
      Vector location, Vector typepage, Vector source){
    boolean createdfile=false;
    while(!location.isEmpty())
      {
File fl=(File)location.firstElement();
directory.remove(0);
source.remove(0);
location.remove(0);
typepage.remove(0);
try{
  FileOutputStream fos = new FileOutputStream(fl);
  PrintStream ps= new PrintStream(fos);
  ps.println("This is the end of your query...");
  ps.close();
}
catch(IOException io){System.out.println(io);}

      }
  }

  /**
   * The method that does the deletion. It first makes
   * sure that thes pecified file or directory is
   *  deleteable before attempting to delete it.
```

```
   * If there is a problem, it throws an
   * IllegalArgumentException
   * @param File the file to delete
   */
  public void delete(File f) {

    // Make sure the file or directory exists and isn't
      write protected
    if (!f.exists()) System.out.println(
"Delete: no such file or directory: " );
    if (!f.canWrite()) System.out.println(
"Delete: write protected: ");
    // If it is a directory, make sure it is empty
    if (f.isDirectory()) {
      String[] files = f.list();
      if (files.length > 0) System.out.println(
  "Delete: directory not empty: ");
    }
    // If we passed all the tests, then
    attempt to delete it
    boolean success = f.delete();
    // And throw an exception if it didn't
    work for some (unknown) reason.

    if (!success) System.out.println(
"Delete: deletion failed");
  }
}
```

C.2 GetWebFile

```
package greenstonetoavantgo.core;

import greenstonetoavantgo.chi.*;
import java.io.*;
import java.net.*;
import java.lang.*;
import java.net.Authenticator;
import java.net.PasswordAuthentication;

/**
This class has for goal the download of a webfile
*/
public class GetWebFile
{
  private MyGuiAuthentification gui;
/*
We use two private field: file is the result
(the webfile )and encoding is how this page
is encoding ASCII,etc
*/
  private File file;
  private String encoding;

  private String login;
  private char[] passwd;
  private String cach;

  public GetWebFile(String l,char[] p,String c){
  login=l;
  passwd=p;
  cach=c;
  }

    /* This method allow to get the result */
  public File getFile(){
    return this.file;
  }
    /* This method allow to get the encoding of the
```

```
      webfile */
  public String getEncoding(){
    return this.encoding;
  }
    /** This is the main method of this class,
it start the download. It receives an url like
http://www.nzdl.org/index.html and an
existing empty file who will be the
perfect copy of the web file.
The result is true if the file has
been successfully downloaded. */

  public boolean work(String url,File fl){
    this.file=fl;
    URL urlObj;
    BufferedReader br=null;
    boolean ntexst=false;
    try{
      urlObj=createConnection(url);
      br=createStream(urlObj);
      downloadFile(br);

    }catch(IOException e){System.out.println(
"Error during GetWebFile : " + e);}
    return ntexst;
  }
  private URL createConnection(String url)
      throws IOException{
    URL urlObj;
    urlObj=new URL(url);
    return urlObj;
  }
  private BufferedReader createStream(URL urlObj)
      throws IOException{

    URLConnection urlC= urlObj.openConnection();
    System.getProperties().put( "proxySet", "true" );
    System.getProperties().put("http.proxyType","4");
    System.getProperties().put( "proxyHost", cach);
    System.getProperties().put( "proxyPort", "8080" );
    Authenticator.setDefault(
```

```
new MyAuthentification(login,passwd));


    this.encoding=urlC.getContentType();
    if(encoding!=null){
      int j=encoding.indexOf("charset=");
      this.encoding=this.encoding.substring(j+8);//trim
    }
    InputStream in = urlObj.openStream();
    BufferedReader br = new BufferedReader(
new InputStreamReader(in,encoding));
    return br;
  }

  private void downloadFile(BufferedReader br)throws IOException{
    FileOutputStream fos = new FileOutputStream(this.file);
    Writer w=new BufferedWriter(new OutputStreamWriter(fos,this.encoding));
    PrintWriter pw=new PrintWriter(w);
    String ligne="";
    while((ligne=br.readLine())!= null){
      pw.println(ligne);
    }
    pw.close();
  }
}
```

C.3 CgiBinLibrary

```
package greenstonetoavantgo.core;

import greenstonetoavantgo.*;
import java.io.*;
import java.lang.*;
import java.util.*;
import java.net.*;

/**
 * This class analyse all the urls
 */
public class CgiBinLibrary
{
  private String line;
  private String  frme="#frame#no";

  private void setFrame(String b){
    this.frme=b;
  }
  public String getFrame(){
    return this.frme;
  }
  public String getNewLine(){
    return this.line;
  }

    /**
     * This method receive the name of the current directory
     * and the name of the file
     * and his create a string that it will be the relative
     * url embed in the new webfile.
     */

  private String relativeDir(String rcur,String ra0){
    rcur=rcur.replace(java.io.File.separatorChar,'/');
    ra0=ra0.replace(java.io.File.separatorChar,'/');
    String rdir="";
    if(rcur!=ra0){
```

```
      StringTokenizer ra0ST=new StringTokenizer(ra0,"/");
      StringTokenizer rcurST=new StringTokenizer(rcur,"/");
      int rla=ra0ST.countTokens();
      int rlcur=rcurST.countTokens();

      String rc=rcurST.nextToken("/");
      String ra=ra0ST.nextToken("/");
      boolean more=rcurST.hasMoreTokens();
      while((rc==ra)&&(more)){
rc=rcurST.nextToken("/");
ra=ra0ST.nextToken("/");
      }
      if(rlcur<rla){
if((rc!=ra)){
  rdir="../";
  ra=ra+'/';
  while(ra0ST.hasMoreTokens()){
    ra=ra+ra0ST.nextToken("/")+'/';
  }
  while(rcurST.hasMoreTokens()){
    rcurST.nextToken();
    rdir=rdir+"../";
  }
  rdir=rdir+ra;
}
else{
  while(ra0ST.hasMoreTokens()){
    rdir=rdir+ra0ST.nextToken("/")+'/';
  }
}
      }else{
rdir="../";
ra=ra+'/';
while(ra0ST.hasMoreTokens()){
  ra=ra+ra0ST.nextToken("/")+'/';
}
while(rcurST.hasMoreTokens()){
  rcurST.nextToken();
  rdir=rdir+"../";
}
rdir=rdir+ra;
```

```
      }
    }

    return rdir;
  }
    /** This method replace the line containing the old url by
     *  the new one.
     *
     * @param line it's the line that should be replaced
     * @param cur it's the new name of the the url
     * @return array which contains the first place a[0] the
     *   relative directory where the web file is stored, and
     * a[1] the name of the file.
    */
    public String[] replace(String line,String cur){
      /* There is some pages that have frame instead of a real
structure build by Greenstone. By default,
frame is no. During the analyse of the cgi we can see
the contrary. */

    setFrame("#frame#no");
    /*This is the time when we assign the value of the line to
      the private field. We will still work
      with this line */
    this.line=line;
    String[] a=null;
    StringBuffer lb=new StringBuffer(line);

    //try to find if this line has an url
    int begin=line.indexOf("href=\"");
    int end=-1;
    //if yes we enter in loop, until there isn't any url to be
    //parsed.

    while(begin!=-1){
      begin+=6;
      end=line.indexOf('\"',begin);
      //extract from the line the proper url
      String url=line.substring(begin,end);
      /* give this url to "urlAnalyse", wich will give me an
array with at the first place a[0] the
```

```
      relative directory where the web file is stored, and
      a[1] the name of the file.*/
      a=urlAnalyse(url,false);
      if(a!=null){
  /* The method relativeDir take as paramater cur which
     is the current directory.This method will
     create a relative url for the new file like
     ../../../directory/file.html. For instance,
     if the file has as a pathname
     C:\mycomputer\directory\mydirectory\file-1.html and
     if the url
     embed in this file has \mydirectory\my\file-1-1.html
     thus the relative ulr embed in file-1.html
     will be my\file_1.html. The current directory is
     C:\mycomputer\directory\mydirectory\
  */
String p=(relativeDir(cur,a[0])+a[1]).replace(
    java.io.File.separatorChar,'/');
this.line=(lb.replace(begin,end,p)).toString();
      }
      //line.indexOf("href",end);  //Attention pour gerer
      plusieur adresse sur la meme ligne
      begin=-1;
    }
    // Here we verify if there is a frame embed in the file
    int bf=line.indexOf("<frame");
    int ef=-1;
    while(bf!=-1){
      bf=line.indexOf("src=");
      if(bf!=-1){
bf+=5;
ef=line.indexOf('\"',bf);
String url=line.substring(bf,ef);
a=urlAnalyse(url,true);
if(a!=null){
  String p=(relativeDir(cur,a[0])+a[1]).replace(
      java.io.File.separatorChar,'/');
  this.line=(lb.replace(bf,ef,p)).toString();
}
      }
      bf=-1;
```

```
    }
    return a;
  }

  private String[] urlAnalyse(String url,boolean frame){

    String[] a=null;
    //if(url.endsWith())
    if(frame){
// if this is an outside site. This is not yet
//implemented
      if(url.indexOf("http://")!=-1)
{
  //initialisation(url);
}else{
  a=cgiAnalyse(url);
}
    }else{
  a=cgiAnalyse(url);
    }
    return a;
  }

    /**
    *This method analyse the cgi in the url and give
    *the argument of this cgi.
    *@param String the url to be analyzed
    *@return array containing in a[0] the directory
    *            a[1] the file
    *            a[2] the url of the next file to
    *            download
    */
  private String[] cgiAnalyse(String url){

    String[] a=null;;
    String cgi=url;
    /* If the cgi have this argument f=1 that means
       that there is a frame. After that, we just have to
       to remove this from the url. It's fine because
       we will find this web page without the frame.
       Actually,in this case the page is divided in 2.
```

```
       First, the navigation bar and then the web page.
       When, you remove the &f=1, you'll just find the
       web page without the navigation bar.
    */
    int f=cgi.indexOf("&f=1&");
    if(f!=-1){
      String b=cgi.substring(0,f);
      String e=cgi.substring(f+4);
      url=b+e;
      setFrame("#frame#yes");
    }
    /* Would like to see if this is document. Otherwise
       we don't care about this page. After this argument
     we will find the name of the file.*/

    int i=cgi.indexOf("&a=d&");
    if(i!=-1){
/* Some urls finishing with .pr argument, this is a
   way provide by the cgi to go up in the previous
   page without the need to rename it. For instance,
   file.1.2.pr will mean file.1.2. We do that
   otherwise we will have two same file with a
   different name.
*/
      int pr=cgi.indexOf(".pr");
      if(pr!=-1){
cgi=cgi.substring(0,cgi.lastIndexOf('.',pr-1));
      }
      i+=5;
      /*The name of the file will start after the "cl=".
It might have something between those two arguments
       but we don't care.*/
      i=(cgi.indexOf("cl=",i))+3;
      int j=cgi.indexOf('&',i);
      if(j!=-1){

a=createNameFile(cgi.substring(i,j),
 cgi.substring(j+3));
      }
      else{
```

```
a=createNameFile(cgi.substring(i));
      }
    }

    if(a!=null)a[2]=url;
    return a;
  }

  /** Here we create the name file from a cgi arguments,
      for instance if the name of the file has as a name
  * HASH136986356998314.3.2 then we'll convert into
  HASH136986956998314/3/ for the directory and 2.html as
  * the file.
  * @param String the cl argument of the cgi-bin
  * @return a[0] the directory
  * a[1] the file
  */
  private String[] createNameFile(String cl){
    String[] a={"","","",""};
    a[0]=cl.replace('.',java.io.File.separatorChar);
    a[1]=(a[0].substring(a[0].lastIndexOf(
java.io.File.separatorChar)+1)).concat(".html");
    return a;
  }
  /**
  * Here we create the name file from a cgi arguments,
  *for instance if the name of the file has as a name
  * HASH136986356998314.3.2 then we'll convert into
  *HASH136986956998314/3/ for the directory and 2.html as
  * the file.
  * @param String the cl argument of the cgi-bin
  * @param String the d argument of the cgi-bin
  * @return  a[0] the directory
  * a[1] the file
  */
  private String[] createNameFile(String cl, String d){
    String[] a={"","","",""};
    int k=d.indexOf('.');
    if(k==-1){
      a[0]=(cl.replace('.',java.io.File.separatorChar))
  +java.io.File.separatorChar+d;
```

```
      a[1]=d.concat(".html");}
    else{
      a[0]=(cl.replace('.',java.io.File.separatorChar))
  +java.io.File.separatorChar+d.substring(0,k);
      a[1]=(d.replace('.','_')).concat(".html");}
    return a;
  }
  /** This is the method that allow the initialisation,
   *of the project
   * @param String the project starting point
   * @parm String the directory to store the project result
   * @param a[0] the directory
    String url=lt[2];
    String dir=localDirectory+lt[0];
   * a[1] the file
   * a[2] the url
   * a[3] is the place on the server where is stored
   *the files
  */
  public String[] initialisation(String url,String l,
      String login, char[] pass, String cach){
    String[] a=null;
    GetWebFile GWF=new GetWebFile(login, pass, cach);
    String server="";
    String linit="";
    int ind=-1;
    File d=new File(l);
    File fl=new File(l+java.io.File.separatorChar+"init.tmp");
    if(url.startsWith("http://")){
      try{
d.mkdir();
fl.createNewFile();
GWF.work(url,fl);

FileInputStream fis = new FileInputStream(GWF.getFile());
BufferedReader br=new BufferedReader(
    new InputStreamReader(fis));
linit=br.readLine();
ind=linit.indexOf("href=\"");
while((ind==-1)){
  linit=br.readLine();
```

```
  ind=linit.indexOf("<a href=\"");
  if(ind!=-1){
    if(linit.indexOf("http://",ind)!=-1){
      ind=-1;
    }
  }
}
fl.delete();
System.out.println(ind);
      }
      catch(IOException e){System.out.println(
  "Error during initialisation..."+e);}
      String ref=linit.substring(ind+9);
      System.out.println(ref);
      StringTokenizer ST=new StringTokenizer(ref,"/");
      ref=ST.nextToken();
      ind=url.indexOf(ref);
      server=url.substring(0,ind-1);
      System.out.println(server);
      a=urlAnalyse(url,false);
      if(a!=null){
a[0].replace('.',java.io.File.separatorChar);
a[3]=server;
      }
    }

    return a;
  }
}
```

C.4 WebFileFactory

```
package greenstonetoavantgo.core;

import greenstonetoavantgo.*;
import java.io.*;
import java.lang.*;
/**
 * This class format the web file
 */

public class WebFileFactory{

  private File fl;
  private BufferedReader br;
  private PrintWriter pw;
  private String l;
  /**
   * Constructor
   *@param File the file to format
   *@param String his encoding
   *@param Sring the type of this file
   */

  public File formatFile(File file,
   String encoding, String type){
    try{
      File temp=createFileStream(file,encoding);
      pw.println("<!--begin-->");
      if(type.equals("bshelf.gif#frame#no")){
replaceHead();
replaceBody();
automatique();
      }
      if((type.equals("clsdfldr.gif#frame#no"))
 ||(type.equals("openfldr.gif#frame#no"))
 ||(type.equals("book.gif#frame#no"))){
replaceHead();
replaceBody();
removeUntil("<center>");
```

```
formatBook();
}
      if(type.equals("itext.gif#frame#yes")){
simpleFormat();
      }
      if(type.equals("itext.gif#frame#no")){
pw.println("<!-- itext.gif#frame#no -->");
replaceHead();
replaceBody();
removeUntil("<center>");
formatTxt();
      }
      if((type.equals("index"))||
 (type.equals("unknown#frame#yes"))
 ||(type.equals("unknown#frame#no"))){
replaceHead();
replaceBody();
automatique();
      }else{
simpleFormat();
      }
      boolean b=temp.delete();
      System.out.println("for "+b);

      pw.close();
    }catch(IOException er){
System.out.println(
    "Error during WebFileFactory(): "+er);}
    return this.fl;
  }
  /**
   * Replace the head in the file.
   */
    private void replaceHead()throws IOException{
    l=br.readLine();
    while(l.indexOf("</title>")==-1){
      pw.println(l);
      l=br.readLine();
    }
    pw.println(l);//write the sentence
    pw.println("<META NAME=\"HandheldFriendly\"
```

```
    content=\"True\">  <!--GTA: add --> ");
    while(l.indexOf("</head>")==-1)
l=br.readLine();//skip all is on the head
    pw.println(l);//print the head
    pw.println("<!-- GTA:end of replace head -->");
  }
  /**
   * Replace the body
   */
  private void replaceBody()throws IOException{
    boolean stop=false;
    l=br.readLine();
    while((l!=null)&&(!stop)){
      if(l.indexOf("<body")!=-1){
br.readLine();
pw.println("<body>");
      stop=true;
      }else{
pw.println(l);
l=br.readLine();
      }
    }
    pw.println("<!--sam:end of replaceBody -->");
  }
/**
 * Launch a simple format in case the type cannot
 * be determined
 */
  private void simpleFormat()throws IOException{
    l=br.readLine();
    while(l!= null){
      int begin=l.indexOf("<table width=");
      if(begin!=-1){
int end=l.indexOf(">",begin);
StringBuffer lb=new StringBuffer(l);
l=(lb.replace(
    begin,end+1,"<table>")).toString();
      }
      pw.println(l);
      l=br.readLine();
      //seulement les tables et les images
```

```
    }
  }
  /**
   * Format file containing a list of books
   */
  private void formatLstBook()throws IOException{
    pw.println(
"<!-- GTA: begin of formatLstBook() -->");
    br.readLine();
    br.readLine();
    removeBanner();
    removeNavigation();
    //removeUntil("<p>");
    pw.println("<table><tr><td><!-- sam : add -->");
    simpleFormat();
  }
  /**
   * Format file containing a book whit its table
   *of matters
   */
  private void formatBook()throws IOException{
    pw.println("<!-- GTA: begin formatBook() -->");
    //removeObj();
    removeFctExcl();
    br.readLine();
    br.readLine();
    br.readLine();
    br.readLine();
    br.readLine();
    removeNavigation();
    removeFctIncl(true);
    simpleFormat();
  }
  /**
   * Formating a file containing a book
   * but keep just the text of this book not
   * the table of matters
   */
  private void formatTxt()throws IOException{
    pw.println(
"<!-- GTA: begin of formatTxt() -->");
```

```
    // removeObj();
    //removeFct();
    //removeNavigation();
    removeFctExcl();
    br.readLine();
    br.readLine();
    br.readLine();
    br.readLine();
    br.readLine();
    removeNavigation();
    br.readLine();
    removeFctIncl(false);
    removeUntil("<p>");
    simpleFormat();
  }


  private void removeObj()throws IOException{
    boolean stop=false;
    l=br.readLine();
    while((l!=null)&&(!stop)){
      if(l.indexOf("<center")!=-1){
while(l.indexOf("</center")==-1)
    l=br.readLine();
stop=true;
      }else{
pw.println(l);
l=br.readLine();
      }
    }
  }
  private void removeFctExcl()throws IOException{
    pw.println("<!-- GTA: begin removeFct()-->");
    boolean stop=false;
    //l=br.readLine();
    while((l!=null)&&(!stop)){
      if(l.indexOf("<a href=\"")!=-1){
while(l.indexOf("</table>")==-1)
    l=br.readLine();
stop=true;
      }else{
```

```
l=br.readLine();
      }
    }
  pw.println("<!-- GTA: end removeFct()-->");
  }
   private void removeFctIncl(boolean nottxt)
       throws IOException{
    pw.println("<!-- GTA: begin removeFct()-->");
    boolean stop=false;
    //String l=br.readLine();
    while((l!=null)&&(!stop)){
      if(l.indexOf("<a href=\"")!=-1){
//pw.println(l);
while(l.indexOf("</table>")==-1)l=br.readLine();
if(nottxt){
  pw.println(l);}
stop=true;
      }else{
l=br.readLine();
      }
    }
  pw.println("<!-- GTA: end removeFct()-->");
  }
 private void removeUntil(String obj)throws IOException{
   pw.println("<!-- GTA:start removeUntil -->");
    boolean stop = false;
    l=br.readLine();
    while((l!=null)&&(!stop)){
      if(l.indexOf(obj)!=-1){
stop=true;
br.readLine();
      }
      l=br.readLine();
    }
  pw.println("<!-- GTA:end removeUntil -->");
  }
/**
 * Remove the navigation bar
 */
  private void removeNavigation()throws IOException{
  pw.println("<!-- GTA:begin removeNavigation -->");
```

```
    boolean stop = false;
    l=br.readLine();
    while((l!=null)&&(!stop)){
      if(l.indexOf("<nobr>")!=-1){
while(l.indexOf("</nobr>")==-1)l=br.readLine();
stop=true;
      }else{
pw.println(l);
l=br.readLine();
      }
    }
    pw.println("<!-- GTA:end removeNavigation -->");
   /*<!-- Navigation Bar -->
      <!-- End of Navigation Bar -->
    */
  }
  /**
   * Remove the banner
   */
  private void removeBanner()throws IOException{
    boolean stop=false;
    l=br.readLine();
    while((l!=null)&&(!stop)){
      if(l.indexOf(
    "<!-- page banner (_style:pagebanner_) -->")!=-1){
  while(l.indexOf
("<!-- end of page banner -->")==-1)
      l=br.readLine();
stop=true;
      }else{
pw.println(l);
l=br.readLine();
      }
      pw.println("<!-- GTA:end of removeBanner -->");
    }
  /*<!-- page banner (_style:pagebanner_) -->
      <!-- end of page banner -->*/

  }
  /**
   * Add a new navigation bar
```

```
   */
  private void addNavigation(String namefile)
      throws IOException{
    pw.println("<a name=\"top\"></a>");
    pw.println("<table><tr><td><a href=\""
     +namefile+"#bottom\">bottom</a></td>
</tr></table><lr>");
  }
  private void addBottom(String namefile)
      throws IOException{
    pw.println("<BR><table><tr><td><a href=\""
   +namefile+"#top\">top</a></td></tr></table>");
    pw.println("<a name=\"bottom\"></a>");
  }
  /**
   * Try to determine automatically the type
   * of the file just with the structure
   * of the HTML
   */
  private void automatique()throws IOException{
    //body et tout le toin toin
    int type=pageType();
    if(type==1){
      formatLstBook();
    }else{
      if(type==2){
formatBook();
      }else{
simpleFormat();
      }
    }
  }
  private File createFileStream(
      File file,String encoding)throws IOException{
    String fname=file.getAbsolutePath();
    //System.out.println(fname);
    File tmp=new File(fname+".fortmp");
    file.renameTo(tmp);
    file.delete();
    this.fl=new File(fname);
    boolean b= this.fl.createNewFile();
```

```
    System.out.println(b);

    FileOutputStream fos = new FileOutputStream(this.fl);
    Writer w=new BufferedWriter(
new OutputStreamWriter(fos,encoding));
    /*PrintWriter*/this.pw=new PrintWriter(w);

    FileInputStream fis = new FileInputStream(tmp);
    /* BufferedReader*/ this.br=new BufferedReader(
new InputStreamReader(fis,encoding));
    return tmp;
  }
/**
 * When the page type is deteminized it launchs
 *the appropriate treatment.
 */
  private int pageType()throws IOException{

    pw.println("<!-- GTA:begin of determine page -->");
    boolean prems=true;
    int typepage=0;
    String l=br.readLine();
    while(prems){
      if(l.indexOf("table")!=-1){
typepage=1;
prems=false;
      }
      if(l.indexOf("center")!=-1){
typepage=2;
prems=false;
      }
      pw.println(l);
      l=br.readLine();
    }
    pw.println(l);
    pw.println("<!-- GTA:end of determine page -->");
    return typepage;
  }

}
```